PROCRAFT: INSTRUCTION-DRIVEN MUSIC PRODUCTION FROM FREE-FORM TEXT

Anonymous authors
Paper under double-blind review

ABSTRACT

Music production typically involves multiple editing operations that each require specialized tools and expertise. Existing text-guided music editing systems support few operations or instruments, operate on short clips, or handle only one operation per instruction, unlike real workflows where several changes are requested at once. Here, we present ProCraft, a lightweight 0.5B-parameter autoregressive codec language model that edits 30s music excerpts end-to-end from free-form instructions, including compound requests. ProCraft supports seven operations across eight instruments, from tasks with a unique target (e.g., track removal) to creative ones (e.g., accompaniment generation). We align intermediate representations with pretrained music embeddings and pitch and instrument targets to preserve content outside the edit, and gated cross-attention in every decoder layer to keep generation tied to the instruction. To train on compound requests, we introduce On-the-fly Editing-pair Synthesis, which generates paired instructions and audio combining track-level operations with effect changes. ProCraft performs competitively against task-specific systems on single-task editing, and outperforms four free-text editors on compound instructions for track removal, extraction and denoising on real (MoisesDB) and synthetic (Slakh) recordings. In a user study (16 listeners), ProCraft was rated highest on removal and denoing and on-par with a 16-times larger model on extraction and instrument transfer. Ablations further show that representation alignment and cross-attention reduce leakage from unwanted tracks and raise editing success rates, especially on out-of-domain data.

1 INTRODUCTION

Editing a music recording often requires changing both what is played and how it sounds. Task-specific models have made individual operations increasingly capable, including source separation (Rouard et al., 2023; Liu et al., 2025b; Bie et al., 2025), accompaniment generation (Mariani et al., 2024; Rouard et al., 2025), instrument transfer (Tsai et al., 2024; 2025), and audio effects (Doh et al., 2026), but each is designed for a single type of edit, whereas production workflows typically request several changes at once.

Text-guided editing lets users express such requests directly in natural language, yet existing music editors remain limited in scope (Table 1): most cover only a handful of operations or instruments, do not accept free-form text, operate on clips of a few seconds, or perform one operation per instruction. For example, recent models such as Audio-Omni (Tian et al., 2026) edit full mixtures from free-form text but struggle with stem-level operations such as removing a single instrument, whilst LLM2Fx-Tools (Doh et al., 2026) handles multi-step instructions but predicts audio effect parameters rather than generating edited audio.

Performing diverse edits from free-form text in a single generative model raises two challenges. First, the model must infer (possibly several) target stems and operations from loosely worded instructions rather than from fixed options. Second, it must preserve everything the instruction does not target. This is especially hard for creative operations such as accompaniment generation, where the model must generate new material while preserve context from the rest of the mix faithfully. One promising direction is unified audio-text codec language models. In speech, SpeechX (Wang et al., 2024) reads a text query and audio as a single stream to perform tasks such as noise suppression and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Supported music editing operations per system. ● = supported, ▲ = partial or indirect, × = not supported; judgments argued in Appendix H.

Model	Free phrasing	Remove	Extract	Accomp.	Cover	Instr. transfer	Denoise	Effects
ProCraft (ours)	●	●	●	●	●	●	●	●
Instruct-MusicGen (Zhang et al., 2025b)	×	●	●	●	×	×	×	×
Audio-Omni (Tian et al., 2026)	●	×	×	●	●	●	×	×
AudioSep (Liu et al., 2025b)	▲	▲	●	×	×	×	▲	×
SD-Codec (Bie et al., 2025)	×	×	▲	×	×	×	●	×
MSDM (Mariani et al., 2024)	×	×	×	●	×	×	×	×
MusicGen-Stem (Rouard et al., 2025)	▲	×	×	●	×	×	×	×
Pop2Piano (Choi & Lee, 2023)	×	×	×	×	●	×	×	×
PiCoGen2 (Tan et al., 2024a)	×	×	×	×	●	×	×	×
AP-Adapter (Tsai et al., 2024)	●	×	×	▲	▲	▲	×	×
MuseControlLite (Tsai et al., 2025)	●	×	×	▲	▲	●	×	×
ZETA / DDPM-inv. (Manor & Michaeli, 2024)	●	▲	×	×	▲	●	×	▲
SteerMusic. (Niu et al., 2026)	●	▲	×	×	▲	▲	×	▲
LLM2Fx-Tools (Doh et al., 2026)	●	×	×	×	×	×	×	●

target-speaker extraction. Music, however, is harder: It contains several concurrent stems rather than one dominant source; its creative edits must respect harmonic and rhythmic content rather than only remove or isolate existing sounds, and its longer excerpts place the instruction and source further from later output frames.

Here, we propose ProCraft, a 0.5B-parameter autoregressive codec language model initialized from the music generation model InspireMusic (Zhang et al., 2025a). Given a 30 s mixture and a free-form instruction, ProCraft generates the edited audio in an end-to-end manner. It supports seven edit types across eight instrument families, applied alone or together: stem removal, stem extraction, accompaniment generation, single-instrument cover, instrument transfer, denoising, and audio effects. In particular, we introduce two mechanisms to address the challenges above. A *persistent edit head* summarizes the instruction and source audio and is attended by every decoder layer through gated cross-attention (Alayrac et al., 2022) to ensure instruction accessibility over long sequences. *Intermediate-layer music alignment* aligns an intermediate layer with pretrained music embeddings (MERT (Li et al., 2024), StemFX (Cheng et al., 2026)) and with pitch and instrument targets, following representation alignment (Yu et al., 2025), so that the model's representation of the mixture carries the stem-level structure needed to preserve untargeted stems. Because compound editing triplets do not exist at scale, we also introduce On-the-fly Editing-pair Synthesis, in which a tool-calling language model (Yang et al., 2025) writes each instruction and specifies audio effects as executable function calls on multitrack stems.

In summary, our contributions[1] are as follows:

- **A single model for diverse and compound music editing.** We formulate diverse and compound edits from free-form instructions as one task, evaluated on both the requested change and the preservation of untargeted content, so that neither editing the wrong stem nor returning the input unchanged scores well. ProCraft remains competitive with task-specific systems on single operations and outperforms four free-text editors on compound instructions on MoisesDB and Slakh.
- **Two mechanisms for conditioning long codec sequences.** A persistent edit head keeps the instruction accessible to every decoder layer, and intermediate-layer music alignment preserves stem-level structure. Ablations on real recordings show that both reduce leakage from untargeted stems and improve editing success.
- **On-the-fly Editing-pair Synthesis.** A tool-calling language model generates source, instruction, and target triplets for training without manual annotation.

[1] Model weights and code for the model, training, and data generation will be released upon acceptance.

2 PROCRAFT

Given a source mix x and a free-form instruction c, the goal is to use our model f to generate an edited mix $\hat{y} = f(x, c)$ that applies the requested change whilst leaving all other content intact. A codec encoder E maps audio to discrete codes, such that $a^{\text{src}} = E(x)$ and $a^{\text{tgt}} = E(y)$, and t denotes the tokenized instruction. We model the codes of the edited mix autoregressively with f as follows:

$$p_\theta\left(a^{\text{tgt}} \mid t, a^{\text{src}}\right) = \prod_{i=1}^{T} p_\theta\left(a_i^{\text{tgt}} \mid a_{<i}^{\text{tgt}}, t, a^{\text{src}}\right). \tag{1}$$

The model is trained with teacher forcing, and at inference, a^{tgt} is sampled one code at a time and decoded into $\hat{y}$. Importantly, this formulation is shared across all editing operations, from those with a unique target such as track removal to creative tasks such as accompaniment generation.

2.1 BACKBONE, TOKENIZATION, AND SEQUENCE LAYOUT

As illustrated in Figure 1, ProCraft is initialized from the InspireMusic language model (Zhang et al., 2025a), a 0.5B-parameter codec language model pretrained for text-to-music generation on the Qwen-2 architecture (Yang et al., 2024) (d=896, 24 layers). We chose InspireMusic because it already models text tokens and audio codes as a single sequence, which provides a natural starting point for text-guided editing. All weights are fine-tuned, and we add gated cross-attention layers (Section 2.2) and alignment projectors (Section 2.3) to the backbone.

To obtain audio codes, a frozen WavTokenizer codec (Ji et al., 2025) encodes a 24 kHz mono downmix at 75 frames per second into a single stream, such that $a = E(x) \in \{1, \ldots, V\}^T$, with $V = 4096$ and $T = 2250$ for a 30 s clip.Instructions are tokenized using the text tokenizer of the backbone, and the output head predicts over the V audio codes plus one end-of-sequence token.

To place the instruction and audio in the same context, the backbone reads the instruction tokens t, the source codes a^{src}, and K=32 learnable query embeddings q, followed by the target codes, as one causal sequence:

$$s = \left[t\,;\, a^{\text{src}}\,;\, q\,;\, a^{\text{tgt}}\right]. \tag{2}$$

Under causal attention, this ordering determines what each part of the sequence f can access. In particular, the audio source codes are encoded with the instruction already in context, and the queries attend to both the instruction and the source to summarize the requested edit before any target code is produced. These allow the each target code to attend to the full preceding context.

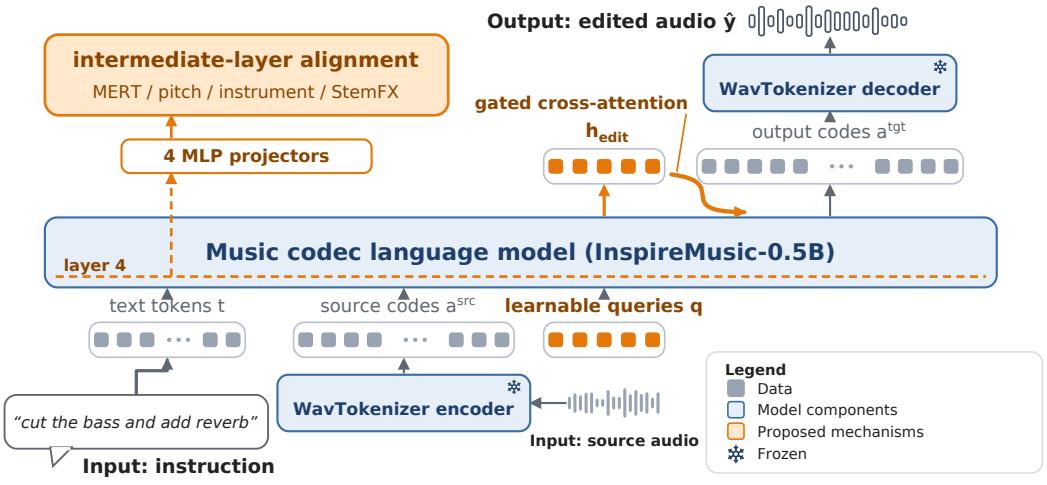

Figure 1: Architecture of ProCraft. The instruction, source codes, and K learnable queries precede the target codes in a single causal sequence. The final hidden states of the queries form the edit head h_{edit}, which is injected into every backbone layer through gated cross-attention. During training, four MLP projectors align layer 4 with MERT, pitch, instrument, and StemFX targets.

At inference, the sampled target codes terminated by the end-of-sequence token are first decoded into waveform using WavTokenizer then upsampled to 48 kHz mono waveform using the frozen flow-matching super-resolution stage and HifiCodec vocoder of InspireMusic (Zhang et al., 2025a). Please see Appendix A for sampling and other implementation details.

2.2 Persistent edit head and gated cross-attention

In the layout of Eq. 2, the instruction appears only at the start, so its share of the causal context shrinks as generation proceeds and its influence on later target codes may weaken. We therefore summarize the requested edit into a few vectors and pass them directly to every decoder layer with the K learnable queries q in Eq. 2. The queries are shared across examples, but because they attend to the instruction and source audio, their final-layer hidden states (the *edit head* $h_{\text{edit}} \in \mathbb{R}^{K\times d}$) describe the edit requested for each input. This follows BLIP-2 (Li et al., 2023), with one difference: BLIP-2 queries replace the image tokens, whereas ours are added alongside the source audio, which remains in full in the context.

Each backbone layer gains a cross-attention path from the target positions to h_{edit}. As in Flamingo (Alayrac et al., 2022), the output of this path is scaled by $\tanh(\alpha)$, with a learnable scalar α initialized at zero, so the model starts from the pretrained backbone and learns during fine-tuning how much to rely on the new path. Because h_{edit} comes from the final layer but is used in every layer, it cannot be computed in the single teacher-forced pass normally used to train codec language models. We therefore train with two passes. The first encodes the prefix $[t\,; a^{\text{src}}\,; q]$ and returns h_{edit} and the prefix key-value cache. The second processes a^{tgt} with teacher forcing, attending to the cache through self-attention and to h_{edit} through cross-attention, so that the generation loss back-propagates into the queries and the prefix. Since each token is processed only once, the added cost is small, and training matches inference, where the prefix is encoded once and h_{edit} is held fixed while target codes are sampled.

2.3 Intermediate-layer music alignment

The edit head keeps the instruction accessible, but editing some stems while preserving the others also requires the backbone to represent the stem-level content of the mix. REPA (Yu et al., 2025) showed that aligning early hidden states of image diffusion transformers with pretrained self-supervised representations speeds up training and improves generation quality. We apply this idea to a codec language model, aligning layer 4 of 24 at the target positions with four targets computed from the target audio: frozen MERT-330M (Li et al., 2024) embeddings, which capture musical structure in mixtures; frame-level pitch and instrument activity, which MERT does not encode explicitly; and StemFX (Cheng et al., 2026) embeddings for audio effects.

Each target has its own MLP projector from the layer-4 hidden states, used only during training. The MERT projector is trained with per-frame cosine distance to the target MERT embeddings ($\mathscr{L}_{\text{mert}}$). The pitch and instrument projectors predict which of 88 pitched-note and 47 drum-note classes, and which of the eight instrument classes, are active at each frame, trained with binary cross-entropy ($\mathscr{L}_{\text{roll}}$, $\mathscr{L}_{\text{inst}}$) on frames with note labels. The effect projector mean-pools the hidden states over the target positions and is trained with cosine distance to the target StemFX embedding ($\mathscr{L}_{\text{fx}}$).

2.4 Training objective

The model is trained with the next-code cross-entropy $\mathscr{L}_{\text{CE}}$ of Eq. 1 over the target codes and the end-of-sequence token. To stabilize training, we further add a z-loss (Chowdhery et al., 2023) that keeps the log partition function of the pre-softmax logits ℓ close to zero, averaged over target positions:

$$\mathscr{L}_z = \left(\log \textstyle\sum_v e^{\ell_v}\right)^2. \tag{3}$$

The final training objective is the weighted sum of these terms and the alignment losses (Section 2.3):

$$\mathscr{L} = \mathscr{L}_{\text{CE}} + \lambda_z \mathscr{L}_z + \lambda_{\text{mert}} \mathscr{L}_{\text{mert}} + \lambda_{\text{p}} \mathscr{L}_{\text{roll}} + \lambda_{\text{i}} \mathscr{L}_{\text{inst}} + \lambda_{\text{f}} \mathscr{L}_{\text{fx}}, \tag{4}$$

with λ_z=10^{-4}, λ_{mert}=0.5, λ_{p}=1.0, λ_{i}=0.5, and λ_{f}=0.5. Appendix D gives the training settings.

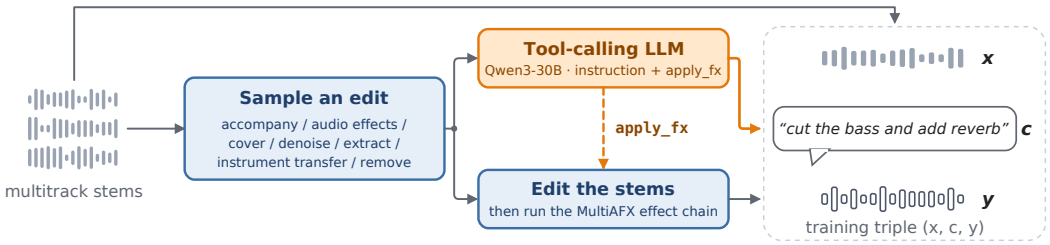

Figure 2: On-the-fly editing-pair synthesis. For each training item, a track-level task and/or audio effects are applied to the stems of a multitrack recording. A tool-calling language model writes the instruction and specifies audio effects as executable function calls, which are rendered on the stems to produce the target mix.

3 On-the-fly editing-pair synthesis

Training a text-guided editor requires triplets of a source recording, an instruction, and the edited result. No existing corpus provides these at scale, particularly for compound instructions that combine track-level operations with audio effects. We therefore synthesize them from multitrack stems (Figure 2): instructions are generated once in advance (see Appendix G.2), and the matching audio is rendered on-the-fly each time an item is drawn during training.

Each audio sample is a random 30 s excerpt from a song in Slakh2100 (Manilow et al., 2019), whose stems and MIDI allow every task to be rendered with a known target. We cover eight instrument families: bass, drums, guitar, piano, organ, brass, reed, and string ensemble. An item combines a track-level task with audio effects (40%), uses a track-level task alone (40%), or uses audio effects alone (20%). Track-level tasks are drawn uniformly from six operations, which together with audio effects make up the seven operations of ProCraft.

For removal and extraction, the target omits or keeps only the requested stem. For accompaniment, the requested instrument is withheld from the source, and the target contains the new part alone. For cover generation, the whole excerpt is re-voiced on a single instrument (guitar, piano, organ, or string ensemble), using a score reduction method for piano covers (Nakamura & Sagayama, 2015) to merge all parts into one playable part. For instrument transfer, the MIDI of one track is re-rendered with the soundfont of the new instrument, so only the timbre changes. For denoising, an environmental sound from ESC-50 (Piczak, 2015), such as rain or an alarm, is repeated throughout the excerpt, and the clean mix is the target. Appendix F gives further details.

Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025) writes each instruction from the sampled task, the instruments present, and one of 30 editing motivations (e.g., "make it sound vintage"; Appendix G), which diversifies both wording and effect choices beyond fixed templates. When effects are included, the model also specifies them as tool calls, which we execute on the stems with MultiAFX (Cheng et al., 2026), following LLM2Fx-Tools (Doh et al., 2026). Because the effects are executed rather than only described, the target matches the instruction by construction.

4 Evaluation

4.1 Metrics

General metrics. *CLAP-T out* and *CLAP-A* are the cosine similarities between the CLAP embeddings (Wu et al., 2023) of the output and, respectively, the instruction and the source audio. *KL* is the divergence between the AudioSet label posteriors predicted by PaSST (Koutini et al., 2022) for the output and the target. *FAD* and *KAD* compare outputs with a fixed reference set of 390 real recordings (240 from MoisesDB, 150 from MUSDB18-HQ) (Rafii et al., 2019; Kilgour et al., 2019; Gui et al., 2024; Chung et al., 2025). These metrics cannot adequately assess editing or preservation: CLAP-A is maximized by returning the source unchanged, and CLAP-T may reward a removal output that still contains the named instrument. We therefore add task-specific metrics for Editing and Preservation.

Operations with a deterministic target. For removal, extraction, denoising and audio effects, we measure Editing and Preservation with *mel-L1* on log-mel spectrograms and multi-resolution STFT distance (*MRSTFT*, four window lengths), and additionally *SI-SDR* for all but audio effects. For removal and extraction, *edit rate* and signed *edit strength* (Zhang et al., 2025b) measure whether, and by how much, the named stem was removed or isolated, using stem energy from HTDemucs-6s (Rouard et al., 2023); we evaluate only drums, bass, guitar and piano, the instruments our baselines support. For audio effects, *AFx-Rep cosine* (Steinmetz et al., 2024) compares the effect style of output and target, and *audio-feature distance* (Vanka et al., 2024) compares their loudness and spectral descriptors. We use AFx-Rep rather than StemFX to avoid circular evaluation, as StemFX is an alignment target (see Section 2.3 on intermediate layer alignment).

Operations without a unique target. Accompaniment and cover generation add new material, and in instrument transfer any realistic timbre of the requested instrument is valid, so comparing with a single reference would penalize valid outputs. For Editing on all three, *instrument match* is the accuracy of PaSST in recognizing the requested instrument, and *isolation strength* tests whether the requested part is output alone rather than within a full mix. For Preservation, *APA* (Grachten & Nistal, 2025) on accompaniment and cover checks, by comparing FADs, that outputs fit their own source better than a randomly paired one; *StructDist* (Liu et al., 2025a) (accompaniment) and *melody chroma accuracy* (Tan et al., 2024b) (cover and transfer) test whether the output follows the source's structure and melody; and for transfer, the *frame-F1* of a MuScriptor transcription (Rouard et al., 2026) against the ground-truth MIDI measures pitch and timing. Details are in Appendix E.

4.2 Test sets and baselines

We evaluate all seven operations on both synthetic and real recordings. Synthetic test items are drawn from the official test split of Slakh2100 (Manilow et al., 2019), and all items from MoisesDB (Pereira et al., 2023). As ProCraft is trained only on Slakh, MoisesDB provides an out-of-domain test on real recordings. Each operation has 150 test items per dataset, each consisting of a 30 s source excerpt, an instruction, and, where applicable, a target.

We compare ProCraft with task-specific systems (AudioSep (Liu et al., 2025b), HTDemucs (Rouard et al., 2023), InstructMusicGen (Zhang et al., 2025b), InstructFX2FX (Yu et al., 2026), MG-Stem (Rouard et al., 2025), MuseControlLite (Tsai et al., 2025), PiCoGen2 (Tan et al., 2024a)) on each single operation, and with four free-text editors (ZETA and a DDIM-inversion editor (Manor & Michaeli, 2024), SteerMusic (Niu et al., 2026), and Audio-Omni (Tian et al., 2026)) on compound instructions. All baselines are run in their native input formats (see Appendix K for checkpoints and settings, and further baselines), and metrics that do not apply to a system are omitted.

5 Results

5.1 Main results

Figure 3 summarizes model performance for each operation (see full results in Tables 3 and 4). Overall, ProCraft was most competitive when the target contains material absent from the source, but trailed dedicated systems when ground-truth target exists.

Creative operations. On accompaniment, cover, and instrument transfer, task-specific systems tended to favor either Editing or Preservation. On Slakh instrument transfer, for example, TokenSynth best preserved pitch and timing (frame-F1 of 0.641) but rendered the requested instrument less reliably (instrument match of 0.420), whereas the DDPM-inversion editor shows the opposite (0.326 and 0.653). ProCraft was the only system above 0.55 on instrument match, frame-F1, and melody chroma. Similarly, piano-cover systems follow the source melody at most slightly more closely than ProCraft (Pop2Piano on Slakh, PiCoGen2 on MoisesDB) but fit their source less well (APA of 0.394–0.712 vs 0.775–0.820). The exception is in accompaniment, where MusicGen-Stem led on instrument match and APA, and AP-Adapter on instrument transfer on MoisesDB.

Audio effects. InstructFX2FX was closest to the processed target in both datasets. Applying DSP rather than regenerating the audio does not fully explain this, however, as ProCraft outperformed the DSP-based FxSearcher and Text2FX on mel-L1, MRSTFT, and AFx-Rep (e.g., AFx-Rep cosine

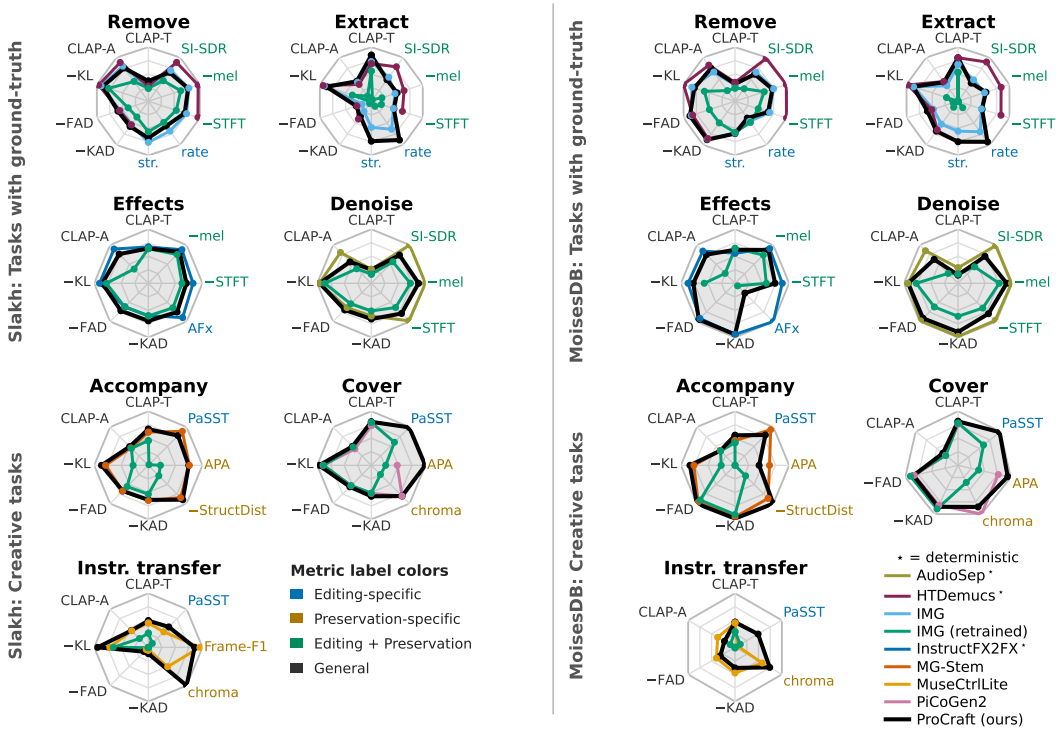

Figure 3: Model performance on single-task music editing from free-form text, tested on synthetic recordings in Slakh (left) and real recordings in MoisesDB (right) over diverse metrics and model baselines. We multiply a metric by -1 if the raw metric value is better when lower so larger radii always indicate better results. See full results in Appendix B.

of 0.195 vs 0.157 and 0.092 on MoisesDB). This suggests that inferring the correct effect from the instruction mattered just as much as the signal processing.

Removal, extraction, and denoising. Here, dedicated source separator models led SI-SDR by a wide margin (e.g., +8.31 vs −2.96 dB for HTDemucs and ProCraft on Slakh removal). To explain this gap, we encoded and decoded the ground-truth targets with WavTokenizer, which yielded +1.05 dB, so 7.3 dB of this 11.3 dB gap stems from the low-bitrate (0.9 kbps) single-codebook codec rather than the model. Nevertheless, SI-SDR does not check whether the edit actually took place: on MoisesDB extraction, AudioSep scored higher than ProCraft (−3.69 vs −15.14 dB) yet isolated the requested stem in only 7.3% of items, whereas ProCraft succeeded in 82.0% of items.

5.2 Ablation Study

We tested the importance of joint instruction-audio conditioning, as well as our cross-attention and alignment mechanisms by retraining Instruct-MusicGen on our data (IMG retrained), and training ProCraft-plain, which removes the gated cross-attention and alignment losses.

Joint conditioning. IMG (retrained) trailed ProCraft on every operation (Figure 3). Whereas Instruct-MusicGen conditions on the instruction and source audio through separate paths (Zhang et al., 2025b), ProCraft reads both in one sequence, so every output code can attend to them together. This suggests improvements due to joint audio-text conditioning and model backbone.

Cross-attention and alignment. ProCraft and ProCraft-plain performed similarly on accompaniment, cover, and instrument transfer, but differed on operations that must remove content whilst keeping the rest intact. On MoisesDB removal, ProCraft raised edit rate from 26.0% to 38.0% and edit strength from 4.9 to 7.4 dB, and on extraction it improved SI-SDR in both datasets (e.g., −15.1 vs −17.4 dB on MoisesDB). Figure 4 further shows that the advantage of ProCraft does not depend

on the 6 dB threshold used to define edit rate, as it led at every threshold in both datasets. These demonstrate the improvement from cross-attention and alignment, particularly on out-of-domain, real datasets.

5.3 Compound edit-and-effects instructions

Production requests often combine an edit with a change in sound, such as "remove the bass and make the piano brighter and more spacious". Here, we paired each of the six track-level operations with audio effects and compared ProCraft with the four free-text editors (see Section 4.2).

Removal, extraction, and denoising. ProCraft outperformed all four editors on Slakh by at least 8.25 dB SI-SDR (Table 2); on removal, for example, it reached −7.39 dB against −24.88 dB for the next best system. ProCraft also led on MoisesDB by at least 6.33 dB.

Table 2: SI-SDR (dB, higher is better) on compound instructions in Slakh.

System	Removal	Extraction	Denoising
ProCraft (ours)	−7.39	−15.73	−4.74
ZETA	−46.18	−38.03	−35.42
DDIM inv.	−44.83	−41.26	−32.18
SteerMusic	−25.60	−27.20	−29.90
Audio-Omni	−24.88	−23.98	−26.02

Accompaniment, cover, and instrument transfer. Each baseline fell short on at least one aspect (Figure 5). Audio-Omni scored higher than ProCraft on APA for accompaniment and cover, but its negative isolation strength shows that it often returned a full mix instead of the requested part. The DDIM-inversion editor isolated the part more strongly on some operations, but fit the source far less well. SteerMusic rarely rendered the requested instrument on accompaniment and barely isolated the part on transfer. ProCraft was the only editor without a large drop on any metric across the three operations, although its isolation on instrument transfer remains imperfect. Appendix C gives full results of this analysis.

5.4 Listening test

We ran a MUSHRA-style listening test (ITU-R, 2015) to compare how listeners judged ProCraft and the four free-text editors on compound instructions in MoisesDB. For each of the six operations, we prepared three recordings with an instruction. Each of the sixteen listeners heard one of these per operation, together with the outputs of all five systems in random order, and rated each output from 0 to 5 on Editing, Preservation, and audio quality. Appendix J gives the protocol, participant details, and statistical tests (task-wise repeated-measures ANOVA with Holm-corrected paired comparisons).

Removal and denoising. ProCraft received the highest mean ratings on all three aspects (Figure 6). It was rated significantly higher than all four editors on every aspect for denoising, and on Preservation and audio quality for removal.

Extraction and instrument transfer. ProCraft and Audio-Omni were rated highest, with no significant difference between them on any aspect. This is notable given that Audio-Omni has 16 times more parameters (7.9B vs 0.5B) and was trained on about 300 times more audio (44,100 vs 145 hours).

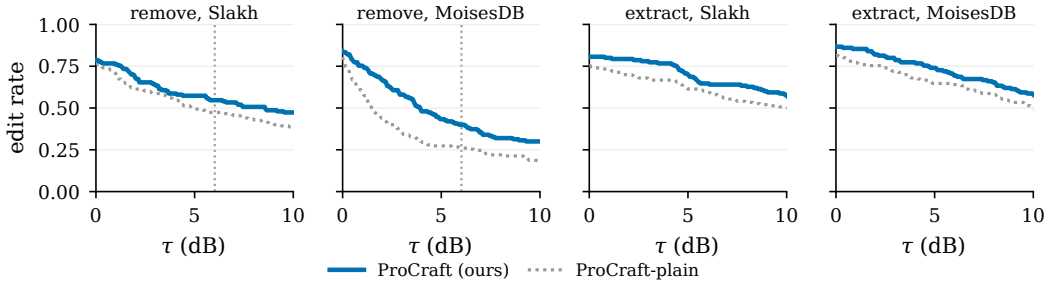

Figure 4: Fraction of items whose edit strength exceeds a threshold τ, for ProCraft and ProCraft-plain (150 items per panel). The dotted line marks the 6 dB threshold used for edit rate.

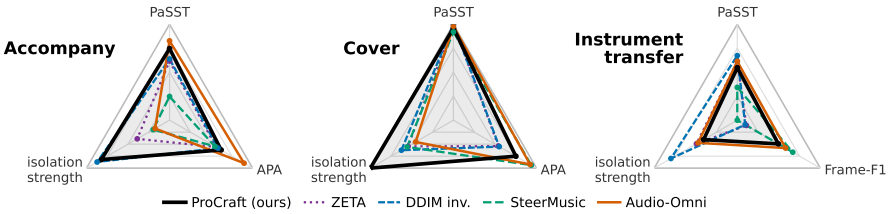

Figure 5: Model performance on music editing from compound instructions on accompaniment, cover, and instrument transfer on Slakh. Larger radii indicate better performance; isolation strength shares the same range across panels. See Appendix C for full results including MoisesDB.

Accompaniment and cover. Audio-Omni and SteerMusic received higher mean Preservation ratings than ProCraft, significantly so on cover. However, both systems often returned the original mix rather than only the requested part (negative isolation strength in Table 6), which would make their outputs sound closer to the source. Audio-Omni also received higher audio-quality ratings on both operations, and although these differences were not significant, they suggest that audio quality is the main area for ProCraft to improve.

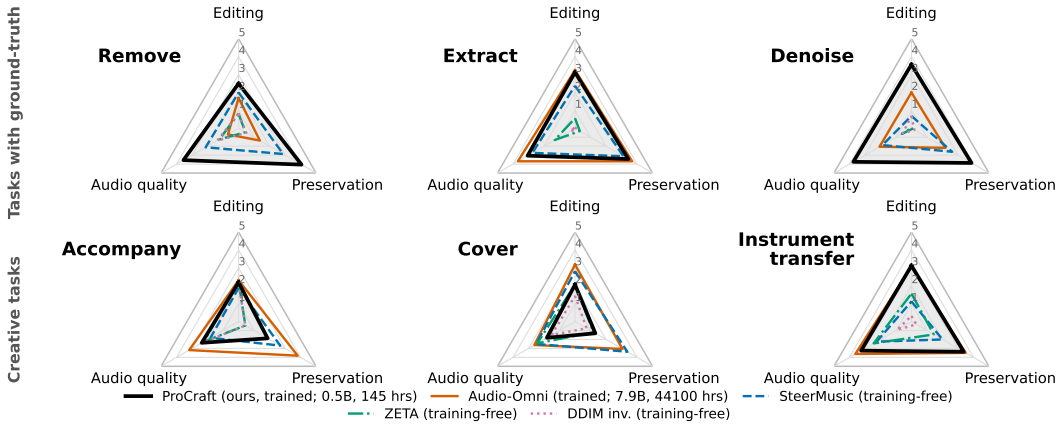

Figure 6: MUSHRA-style user study on perceived audio from music editing with compound instructions. Each triangle shows a task's mean Editing, Preservation, and Audio-quality ratings (0 to 5) from sixteen raters; See Table 9 and Appendix J for results, statistical inference, and protocol details.

6 Conclusion

Music production typically involves several changes to a recording at once, yet existing editors handle only a fixed type of edit or one operation per instruction. Here, we presented ProCraft, a single 0.5B-parameter codec language model that performs seven editing operations across eight instrument families on 30 s excerpts from free-form instructions, including compound requests that combine stem-level operations with audio effects. Trained only on triplets from On-the-fly Editing-pair Synthesis, ProCraft outperformed four free-text editors on compound instructions, and was rated highest by listeners on removal and denoising, and on par with the 16-times larger Audio-Omni on extraction and instrument transfer. Ablations on real recordings showed that the persistent edit head and intermediate-layer music alignment together reduce leakage from untargeted stems and raise editing success rates. Nevertheless, our generative model still trailed behind source-separator models such as HTDemucs in tasks with ground truth (such as track removal), and listeners generally preferred larger models such as Audio-Omni on creative tasks like accompaniment and cover generation. Producing high-quality audio edits from natural language input whilst preserving the source in a lightweight model thus remains an open challenge.

AI USE STATEMENT

A large language model is part of the data pipeline, which is a required disclosure. Qwen3-30B-A3B-Instruct-2507 writes the freely phrased instructions and the typed audio effect chains for every training pair, so the instruction distribution and the audio effect vocabulary both inherit that model's biases.

A large language model also assisted in polishing the text of this paper and in writing code. Every number reported here is produced by that code from the stored model outputs, and each table and figure was checked against the underlying results files. It was not used for the research idea, the experimental design, the choice of what to claim, or the analysis of results, all of which are the authors' own work. The authors take responsibility for all claims in the paper.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, et al. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Xiaoyu Bie, Xubo Liu, and Gaël Richard. Learning source disentanglement in neural audio codec. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

Yuan-Chiao Cheng, Jui-Te Wu, Brian Chen, Yen-Tung Yeh, Yu-Hua Chen, and Yi-Hsuan Yang. StemFX: Learning mixing style representations via autoregressive fx chain prediction on source-separated stems. In *Proceedings of the 27th International Society for Music Information Retrieval Conference (ISMIR)*, 2026.

Jongho Choi and Kyogu Lee. Pop2Piano: Pop audio-based piano cover generation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.

Annie Chu, Patrick O'Reilly, Julia Barnett, and Bryan Pardo. Text2FX: Harnessing CLAP embeddings for text-guided audio effects. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

Yoonjin Chung, Pilsun Eu, Junwon Lee, Keunwoo Choi, Juhan Nam, and Ben Sangbae Chon. KAD: No more FAD! an effective and efficient evaluation metric for audio generation. In *ICML 2025 Workshop on Machine Learning for Audio*, 2025.

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Seungheon Doh, Junghyun Koo, Marco A. Martínez-Ramírez, Woosung Choi, Wei-Hsiang Liao, Qiyu Wu, Juhan Nam, and Yuki Mitsufuji. LLM2Fx-Tools: Tool calling for music post-production. In *International Conference on Learning Representations (ICLR)*, 2026.

Zach Evans, Julian D. Parker, C. J. Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Stable audio open. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

Maarten Grachten and Javier Nistal. Accompaniment prompt adherence: A measure for evaluating music accompaniment systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

Azalea Gui, Hannes Gamper, Sebastian Braun, and Dimitra Emmanouilidou. Adapting frechet audio distance for generative music evaluation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

ITU-R. Method for the subjective assessment of intermediate quality level of audio systems. Technical Report Recommendation BS.1534-3, International Telecommunication Union, 2015.

Shengpeng Ji, Ziyue Jiang, Wen Wang, Yifu Chen, Minghui Fang, Jialong Zuo, et al. WavTokenizer: An efficient acoustic discrete codec tokenizer for audio language modeling. In *International Conference on Learning Representations (ICLR)*, 2025.

Hojoon Ki, Jongsuk Kim, Minchan Kwon, and Junmo Kim. FxSearcher: Gradient-free text-driven audio transformation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2026.

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *Interspeech*, pp. 2350–2354, 2019.

Kyungsu Kim, Junghyun Koo, Sungho Lee, Haesun Joung, and Kyogu Lee. TokenSynth: A token-based neural synthesizer for instrument cloning and text-to-instrument. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh, and Gerhard Widmer. Efficient training of audio transformers with patchout. In *Interspeech*, pp. 2753–2757, 2022.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Yizhi Li, Ruibin Yuan, Ge Zhang, Yinghao Ma, Xingran Chen, Hanzhi Yin, et al. MERT: Acoustic music understanding model with large-scale self-supervised training. In *International Conference on Learning Representations (ICLR)*, 2024.

Haohe Liu, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Qiao Tian, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D. Plumbley. AudioLDM 2: Learning holistic audio generation with self-supervised pretraining. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32, 2024.

Huadai Liu, Jialei Wang, Xiangtai Li, Wen Wang, Qian Chen, Rongjie Huang, et al. MelodyEdit: Zero-shot music editing with disentangled inversion control. In *Proceedings of the 33rd ACM International Conference on Multimedia*, 2025a.

Xubo Liu, Qiuqiang Kong, Yan Zhao, Haohe Liu, Yi Yuan, Yuzhuo Liu, et al. Separate anything you describe. *IEEE Transactions on Audio, Speech and Language Processing*, 33:458–471, 2025b.

Ethan Manilow, Gordon Wichern, Prem Seetharaman, and Jonathan Le Roux. Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019.

Hila Manor and Tomer Michaeli. Zero-shot unsupervised and text-based audio editing using DDPM inversion. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Giorgio Mariani, Irene Tallini, Emilian Postolache, Michele Mancusi, Luca Cosmo, and Emanuele Rodolà. Multi-source diffusion models for simultaneous music generation and separation. In *International Conference on Learning Representations (ICLR)*, 2024.

Eita Nakamura and Shigeki Sagayama. Automatic piano reduction from ensemble scores based on merged-output hidden markov model. In *Proceedings of the International Computer Music Conference (ICMC)*, 2015.

Xinlei Niu, Kin Wai Cheuk, Jing Zhang, Naoki Murata, Chieh-Hsin Lai, Michele Mancusi, et al. SteerMusic: Enhanced musical consistency for zero-shot text-guided and personalized music editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Igor Pereira, Felipe Araújo, Filip Korzeniowski, and Richard Vogl. MoisesDB: A dataset for source separation beyond 4-stems. In *Proceedings of the 24th International Society for Music Information Retrieval Conference (ISMIR)*, 2023.

Karol J. Piczak. ESC: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM International Conference on Multimedia*, pp. 1015–1018, 2015. doi: 10.1145/2733373.2806390.

Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 367–372, 2014.

Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. MUSDB18-HQ - an uncompressed version of MUSDB18. https://doi.org/10.5281/zenodo.3338373, 2019.

Simon Rouard, Francisco Massa, and Alexandre Défossez. Hybrid transformers for music source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

Simon Rouard, Robin San Roman, Yossi Adi, and Axel Roebel. MusicGen-Stem: Multi-stem music generation and edition through autoregressive modeling. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

Simon Rouard, Michael Krause, Axel Roebel, Carl-Johann Simon-Gabriel, and Alexandre Défossez. MuScriptor: An open model for multi-instrument music transcription. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2026.

Christian J. Steinmetz, Shubhr Singh, Marco Comunità, Ilias Ibnyahya, Shanxin Yuan, Emmanouil Benetos, and Joshua D. Reiss. ST-ITO: Controlling audio effects for style transfer with inference-time optimization. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2024.

Chih-Pin Tan, Hsin Ai, Yi-Hsin Chang, Shuen-Huei Guan, and Yi-Hsuan Yang. Picogen2: Piano cover generation with transfer learning approach and weakly aligned data. In *Proceedings of the 25th International Society for Music Information Retrieval Conference (ISMIR)*, 2024a.

Chih-Pin Tan, Shuen-Huei Guan, and Yi-Hsuan Yang. Picogen: Generate piano covers with a two-stage approach. In *Proceedings of the International Conference on Multimedia Retrieval (ICMR)*, 2024b.

Zeyue Tian et al. Audio-Omni: Extending multi-modal understanding to versatile audio generation and editing. *arXiv preprint arXiv:2604.10708*, 2026.

Fang-Duo Tsai, Shih-Lun Wu, Haven Kim, Bo-Yu Chen, Hao-Chung Cheng, and Yi-Hsuan Yang. Audio prompt adapter: Unleashing music editing abilities for text-to-music with lightweight finetuning. In *Proceedings of the 25th International Society for Music Information Retrieval Conference (ISMIR)*, 2024.

Fang-Duo Tsai, Shih-Lun Wu, Weijaw Lee, Sheng-Ping Yang, Bo-Rui Chen, Hao-Chung Cheng, and Yi-Hsuan Yang. MuseControlLite: Multifunctional music generation with lightweight conditioners. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Soumya Sai Vanka, Christian J. Steinmetz, Jean-Baptiste Rolland, Joshua D. Reiss, and George Fazekas. Diff-MST: Differentiable mixing style transfer. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2024.

Xiaofei Wang, Manthan Thakker, Zhuo Chen, Naoyuki Kanda, Sefik Emre Eskimez, Sanyuan Chen, et al. SpeechX: Neural codec language model as a versatile speech transformer. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:3355–3364, 2024.

Karn N. Watcharasupat and Alexander Lerch. A stem-agnostic single-decoder system for music source separation beyond four stems. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2024.

Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Marianna Nezhurina, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations (ICLR)*, 2025.

Song-Ze Yu, Milan Liessens Dujardin, Yuxuan Cai, Wantong Zhang, Brian Cruz, Jeremy Wagner, and Carmine-Emanuele Cella. InstructFX2FX: A multi-turn text-to-effect system for sequential audio effect refinement. In *Proceedings of the 29th International Conference on Digital Audio Effects (DAFx)*, 2026.

Chong Zhang, Yukun Ma, Qian Chen, Wen Wang, Shengkui Zhao, Zexu Pan, et al. InspireMusic: Integrating super resolution and large language model for high-fidelity long-form music generation. *arXiv preprint arXiv:2503.00084*, 2025a.

Yixiao Zhang, Yukara Ikemiya, Woosung Choi, Naoki Murata, Marco A. Martínez-Ramírez, Liwei Lin, et al. Instruct-musicgen: Unlocking text-to-music editing for music language models via instruction tuning. In *Proceedings of the 26th International Society for Music Information Retrieval Conference (ISMIR)*, 2025b.

A IMPLEMENTATION DETAILS

Autoregressive sampling and classifier-free guidance. We sample target codes at temperature 1.0 without top-k or nucleus truncation ($k = 0$, $p = 1.0$). Generation stops at the first audio end-of-sequence token, with a fallback maximum equal to the source-code length: 2,250 decoding steps for a 30 s clip at 75 Hz. Instructions are tokenized to at most 192 text tokens. The prefix is encoded once per guidance branch, and decoding uses a key–value cache. We combine the logits as

$$\ell = \ell_{\varnothing} + 1.5(\ell_c - \ell_{\varnothing}),$$

where c is the instruction-conditioned branch. In the null-instruction branch, instruction tokens are replaced with padding tokens that remain attended, matching instruction dropout during training. Both branches retain the same source audio codes and learnable query embeddings. Each branch computes its own edit-head states from its prefix, so the null-instruction queries still see the source audio; these two edit heads remain fixed throughout target generation.

Gated cross-attention. All 24 decoder layers contain a cross-attention sublayer after the self-attention residual update and before the MLP normalization and MLP. Each sublayer uses 14 heads of width 64, with separate bias-free 896×896 query, key, value, and output projections. The target states receive RMS normalization before supplying the attention queries. A shared memory adapter applies RMS normalization and a bias-free 896×896 linear projection to the 32 edit-head states. Each decoder layer has its own scalar α, initialized to zero, and scales its cross-attention residual by $\tanh(\alpha)$. The cross-attention sublayers and shared adapter add 77,895,576 parameters; the 32 learnable query embeddings add 28,672, for 77,924,248 parameters in total (77.92M). This count excludes the training-only alignment projectors and frozen audio encoders and decoders; the 0.5B designation refers to the backbone.

Alignment projectors and temporal resolution. The projectors read the layer-4 hidden states at target positions. The MERT projector has three linear layers with widths $896 \to 2048 \to 2048 \to 1024$, with SiLU activations after the first two layers. The pitch projector is $896 \to 448 \to 135$ with a GELU activation between the two linear layers. The instrument projector uses the same hidden width, $896 \to 448 \to 8$, with GELU between the two linear layers. For StemFX, target-position states are mean-pooled before a $896 \to 512 \to 512$ projector with GELU. Both WavTokenizer and MERT take 24 kHz mono audio and have a nominal frame rate of 75 Hz. MERT's convolutional strides have a product of 320 samples, matching WavTokenizer's 320-sample hop. Pitch and instrument rolls are constructed from the target MIDI on the 75 Hz codec grid, with cover labels taken from the reduced score used to render the target. StemFX instead takes 44.1 kHz audio grouped into vocals, bass, drums, and other stems. We compute a 512-dimensional embedding for each of three consecutive 10 s segments, average the segment embeddings, and normalize the result to unit length. This is a clip-level target and does not require frame-rate matching.

Super-resolution. The frozen InspireMusic flow model conditions directly on the sampled WavTokenizer codes to generate HifiCodec latents; the HifiCodec generator renders the 48 kHz mono waveform. The length regulator linearly interpolates the encoded conditioning sequence to twice its original number of frames, followed by its convolutional refinement layers. Flow matching uses 10 Euler integration steps. Thus the 48 kHz output is generated through the pretrained latent model and vocoder rather than by waveform resampling alone.

Inference cost. On one RTX 5090, processing a 30 s clip takes approximately 38 s at batch size 1, or 40 s with super-resolution. At batch size 8, the amortized time is approximately 5 s per clip, or 7 s with super-resolution. These times include source encoding, autoregressive generation, and waveform decoding. We use the paper checkpoint in bf16 with CFG scale 1.5 and generate 2,250 codes per clip; each timing is measured on a second run after a warm-up pass.

B FULL RESULTS FOR INDIVIDUAL EDITING TASKS

Tables 3 and 4 give the complete task-specific results behind Figure 3. Their row-wise shading shows the relative range of each metric; it does not make values on different metrics directly comparable. Some metrics are not reported in several cases, where we mark them as –. Instruct-MusicGen returns a full mix on accompaniment, so we separate its added part with HTDemucs-6s before scoring and omit instrument match. Also, as HTDemucs-6s is used to compute edit rate, edit strength, and isolation strength, the model is omitted on those metrics. Piano-cover systems also always output solo piano, so their instrument match is trivially perfect and is thus omitted.

Table 3: Full metric set, **Slakh (synthetic)** domain, N=150 per task. E=editing, P=preservation, E+P=both. Our metrics are defined in Appendix E; a dash marks an undefined metric. **Bold** = best in column, shaded worst to best; circle and star mark paired tests against ProCraft at $p < 0.05$ and $p < 0.01$. IMG is Instruct-MusicGen.

Accompany	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	PaSST^{E} ↑	APA^{P} ↑	StructDist^{P} ↓
ProCraft (ours)	0.200	0.590	0.406	0.476	1.959	0.827	0.614	216.624
ProCraft-plain	0.203	0.581	0.376	0.472	1.937	0.853	0.573	**211.236**
IMG (released)	**0.264***	0.604	**0.374**	0.492	2.005	—	0.565	230.702*
IMG (retrained)	0.134*	0.558*	1.600*	0.549	2.271	0.240*	0.183	262.266*
MSDM	0.143*	**0.614**$^{\circ}$	2.179*	**0.456**	**1.761**	0.493*	0.595	282.190*
MusicGen-Stem	0.183*	0.568°	0.558°	0.475	1.924	**0.913**$^{\circ}$	**0.623**	220.629

Cover	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	PaSST^{E} ↑	APA^{P} ↑	chroma^{P} ↑
ProCraft (ours)	**0.243**	0.593	**0.202**	0.546	2.152	0.993	0.820	0.595
ProCraft-plain	0.241	**0.598**	0.213	**0.543**	**2.141**	**1.000**	**0.822**	0.618
IMG (retrained)	0.239	0.581°	0.373*	0.565	2.327	0.693*	0.218	0.370*
Pop2Piano	0.224*	0.591	0.425*	0.556	2.271	—	0.475	**0.634**$^{\circ}$
PiCoGen2	0.221*	0.549*	0.351*	0.566	2.312	—	0.394	0.594

Instrument transfer	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	PaSST^{E} ↑	Frame-F1^{P} ↑	chroma^{P} ↑
ProCraft (ours)	0.145	0.557	**0.256**	0.834	3.498	0.640	0.558	**0.673**
ProCraft-plain	0.152	**0.560**	0.288	0.856	3.603	0.627	0.546	0.654
IMG (retrained)	0.073*	0.415*	0.852*	0.901	3.841	0.307*	0.071	0.278*
AP-Adapter	**0.173***	0.498*	0.815*	0.856	3.461	0.647	0.311	0.512*
MuseCtrlLite	0.131	0.553	0.720*	0.869	3.676	0.540*	0.614	0.476*
DDPM-inv	0.146	0.551	0.582*	**0.810**	**3.213**	**0.653**	0.326	0.606*
TokenSynth	0.143	0.523*	0.434*	0.890	3.504	0.420*	**0.641**	0.276*

Effects	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	AFx^{E} ↑
ProCraft (ours)	0.193	0.776	0.437	**0.445**	1.792	0.871	1.889	0.393
ProCraft-plain	0.203*	0.774	0.411	0.446	**1.787**	0.857	1.865	0.402°
IMG (retrained)	0.191	0.511*	0.591*	0.510	2.061	1.024*	2.178*	0.350*
InstructFX2FX	0.201	**0.867***	**0.331***	0.512	2.075	**0.743***	**1.458***	**0.452***
FxSearcher	**0.304***	0.505*	0.657*	0.553	1.953	1.467*	2.943*	0.373
Text2FX	0.269*	0.537*	0.519*	0.641	2.672	1.220*	2.327*	0.314*

Remove	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	edit rate^{E} ↑	edit strength^{E} ↑
ProCraft (ours)	0.108	0.774	0.367	0.527	2.143	-2.961	0.975	1.874	0.527	10.153
ProCraft-plain	**0.112**	0.773	0.403	**0.520**	**2.123**	-3.370	0.981	1.881	0.473	7.983°
IMG (released)	0.068*	0.803*	0.298	0.554	2.272	-0.872*	0.944	1.751°	**0.620**	**12.890**$^{\circ}$
IMG (retrained)	0.066*	0.564*	0.599*	0.654	2.779	-12.431*	1.294*	2.452*	0.460	6.267*
AudioSep	0.062*	0.650*	0.332	0.607	2.405	-6.317*	1.500*	2.302*	0.487	9.271
HTDemucs-6s	0.078*	0.859*	**0.198***	0.547	2.255	**8.312***	0.544*	0.973*	—	—
Banquet	0.103	**0.937***	0.205*	0.580	2.443	7.565*	**0.494***	**0.852***	0.480	11.109

Extract	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	edit rate^{E} ↑	edit strength^{E} ↑
ProCraft (ours)	0.258	0.568	0.402	0.719	2.827	-11.346	1.577	2.827	**0.787**	**12.176**
ProCraft-plain	**0.262**	0.573	0.435	0.729	2.885	-12.743*	1.607	2.815	0.707°	10.463
IMG (released)	0.219*	0.553	0.310°	0.791	3.163	-8.488*	1.672°	2.848	0.587*	3.480*
IMG (retrained)	0.165*	0.337*	1.423*	0.890	3.684	-33.954*	2.233*	3.677*	0.160*	-12.369*
AudioSep	0.192*	**0.889***	0.692*	**0.679**	2.967	-3.525*	1.675	2.891	0.067*	-5.802*
HTDemucs-6s	0.208*	0.519*	**0.259***	0.725	**2.642**	**3.356***	**1.280***	**2.239***	—	—
Banquet	0.210*	0.474*	0.410	0.806	3.044	-0.421*	1.671	2.599*	0.620*	8.658°

Denoise	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓
ProCraft (ours)	0.069	0.637	0.223	0.459	1.893	0.804	0.753	1.584
ProCraft-plain	0.069	0.633	0.230	0.457	1.877	0.782	0.777°	1.614°
IMG (retrained)	0.043*	0.516*	0.415*	0.561	2.332	-7.825*	1.089*	2.137*
AudioSep	**0.071**	**0.805***	**0.210**	0.498	2.058	**13.903***	**0.471***	**0.917***
SD-Codec	0.030*	0.651	0.244	**0.451**	**1.832**	8.627*	0.871*	1.971*

Table 4: Full metric set, **real audio** domain, N=150 per task. E=editing, P=preservation, E+P=both. Our metrics are defined in Appendix E; a dash marks an undefined metric. **Bold** = best in column, shaded worst to best; circle and star mark paired tests against ProCraft at $p < 0.05$ and $p < 0.01$. IMG is Instruct-MusicGen.

Accompany	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	PaSST^{E} ↑	APA^{P} ↑	StructDist^{P} ↓
ProCraft (ours)	0.165	0.518	0.472	**0.264**	0.939	0.840	0.366	**209.256**
ProCraft-plain	0.166	0.526	0.469	0.272	0.995	0.840	0.369	216.648°
IMG (released)	**0.257***	0.535°	**0.442**	0.264	**0.933**	—	0.409	217.349°
IMG (retrained)	0.121*	0.513	1.654*	0.323	1.160	0.227*	0.000	262.031*
MSDM	0.119*	**0.629***	1.584*	0.315	1.121	0.560*	0.321	279.847*
MusicGen-Stem	0.134*	0.520	0.622°	0.286	1.022	**0.947***	**0.524**	219.187*

Cover	CLAP-T ↑	CLAP-A ↑	FAD ↓	KAD ↓	PaSST^{E} ↑	APA^{P} ↑	chroma^{P} ↑
ProCraft (ours)	0.246	0.501	0.364	1.331	0.987	**0.775**	0.614
ProCraft-plain	**0.250**	0.504	0.363	1.318	**1.000**	0.751	0.608
IMG (retrained)	0.236	0.455*	**0.326**	**1.202**	0.693*	0.321	0.415*
Pop2Piano	0.210*	**0.569***	0.350	1.338	—	0.712	0.429*
PiCoGen2	0.231*	0.459*	0.374	1.428	—	0.628	**0.672***

Instrument transfer	CLAP-T ↑	CLAP-A ↑	FAD ↓	KAD ↓	PaSST^{E} ↑	chroma^{P} ↑
ProCraft (ours)	0.136	0.411	0.704	2.721	0.607	0.569
ProCraft-plain	0.163*	0.457*	0.716	2.797	0.647	0.570
IMG (retrained)	0.081*	0.327*	0.906	3.805	0.313*	0.275*
AP-Adapter	**0.190***	0.465*	0.748	3.040	**0.653**	**0.632***
MuseCtrlLite	0.132	**0.509***	**0.652**	**2.442**	0.267*	0.505*
DDPM-inv	0.178*	0.480*	0.671	2.560	0.473°	0.549

Effects	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	AFx^{E} ↑
ProCraft (ours)	0.185	0.766	0.598	**0.298**	**1.049**	0.756	1.761	0.195
ProCraft-plain	0.180	0.797*	0.567	0.315	1.112	0.747	1.751	0.202
IMG (retrained)	0.189	0.424*	0.951*	0.503	1.964	1.049*	2.314*	0.118*
InstructFX2FX	0.165	**0.829***	**0.398***	0.316	1.056	**0.680°**	**1.292***	**0.493***
FxSearcher	**0.257***	0.481*	0.811*	0.420	1.375	1.297*	2.679*	0.157*
Text2FX	0.235*	0.459*	0.733	0.578	2.288	1.208*	2.303*	0.092*

Remove	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	edit rate^{E} ↑	edit strength^{E} ↑
ProCraft (ours)	0.091	0.770	0.559	0.344	1.251	-5.114	1.077	2.100	0.380	7.373
ProCraft-plain	0.095	0.804*	0.665*	0.348	1.252	-4.803	1.144°	2.200°	0.260*	4.936*
IMG (released)	0.084	0.711*	0.461°	0.361	1.301	-2.914*	1.028	2.000	0.440	7.017
IMG (retrained)	0.066*	0.422*	0.959*	0.586	2.382	-21.513*	1.430*	2.778*	0.460	6.637
AudioSep	0.072*	0.481*	0.575	0.552	2.046	-11.242*	1.699*	2.926*	0.613*	12.173*
HTDemucs-6s	0.100	0.818*	**0.130***	0.377	1.324	**12.854***	**0.478***	**0.844***	—	—
Banquet	**0.111***	**0.886***	0.170*	**0.330**	**1.165**	11.850*	0.503*	0.887*	**0.713***	**17.550***

Extract	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	edit rate^{E} ↑	edit strength^{E} ↑
ProCraft (ours)	0.242	0.553	0.395	0.524	1.831	-15.137	1.486	2.722	0.820	12.508
ProCraft-plain	0.230	0.558	0.439	0.520	1.861	-17.360*	1.660*	2.937*	0.733°	11.168
IMG (released)	0.205*	0.497*	0.413	0.599	2.295	-11.465*	1.549	2.763	0.633*	5.882*
IMG (retrained)	0.158*	0.257*	1.720*	0.846	3.437	-35.738*	2.689*	4.364*	0.187*	-12.951*
AudioSep	0.185*	**0.928***	0.645*	**0.340**	**1.281**	-3.686*	1.506	2.578	0.073*	-3.944*
HTDemucs-6s	0.243	0.564	**0.185***	0.578	1.916	**8.760***	**0.832***	**1.427***	—	—
Banquet	**0.245**	0.461*	0.268*	0.678	2.347	6.859*	1.268*	1.914*	**0.880**	**21.081***

Denoise	CLAP-T ↑	CLAP-A ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓
ProCraft (ours)	0.039	0.682	0.292	0.319	1.185	-0.765	0.693	1.563
ProCraft-plain	0.043	0.669	0.326	0.307	1.131	-1.679°	0.768°	1.681*
IMG (retrained)	0.044	0.433*	0.596*	0.509	2.042	-14.367*	0.999*	2.086*
AudioSep	**0.080***	**0.836***	**0.280**	0.277	0.947	**13.186***	**0.502***	**0.945***
SD-Codec	-0.010*	0.694	0.381*	**0.264**	**0.892**	7.566*	0.812*	1.983*

C FULL RESULTS FOR COMBINED EDITING AND AUDIO EFFECTS

Tables 5 and 6 give the complete results for the six editing tasks paired with audio effects, one table each for Slakh and MoisesDB. They provide the underlying values and additional metrics for the summaries in Section 5.3.

Table 5: Combined edit and audio effects benchmark, **Slakh (synthetic)** domain, 150 samples per task; the target is the edited and processed audio. E=editing, P=preservation, E+P=both. **Bold** = best in column, shaded worst to best.

Accompany	CLAP-T ↑	CLAP-A in-context ↑	KL ↓	FAD ↓	KAD ↓	PaSST^{E} ↑	APA^{P} ↑	StructDist^{P} ↓	Audio Feat.^{E} ↓	isolation^{E} ↑
ProCraft (ours)	0.170	0.565	**0.681**	0.446	1.782	0.747	0.621	208.784	0.852	7.471
ProCraft-plain	**0.323***	0.553	0.723	0.422	1.677	0.747	0.598	211.537	0.809	**8.880**
IMG (retrained)	0.139*	0.550	1.677*	0.579	2.389	0.193*	0.170	**106.944***	0.945°	-11.929*
ZETA	0.277*	0.605*	1.247*	0.561	2.334	0.617°	0.559	299.294*	0.862	-2.330*
DDIM inv.	0.322*	0.571	1.185*	0.579	2.409	0.638°	0.583	275.083*	0.850	8.607
SteerMusic	0.304*	**0.655***	1.536*	0.534	2.153	0.247*	0.546	117.297*	**0.712***	-6.677*
Audio-Omni	0.126*	0.566	0.931*	**0.368**	**1.377**	**0.827**	**0.897**	153.319*	0.826	-7.204*

Cover	CLAP-T ↑	CLAP-A in-context ↑	KL ↓	FAD ↓	KAD ↓	PaSST^{E} ↑	APA^{P} ↑	chroma^{P} ↑	Audio Feat.^{E} ↓	isolation^{E} ↑
ProCraft (ours)	0.243	0.573	**0.435**	0.493	**1.894**	0.960	0.752	0.574	0.681	11.495
ProCraft-plain	0.305*	0.548*	0.445	0.497	1.897	0.960	0.723	0.592	0.663	**13.896**
IMG (retrained)	0.217°	0.575	1.021*	0.526	2.137	0.227*	0.329	0.191*	0.803*	-7.261*
ZETA	0.400*	0.562	0.892*	0.551	2.249	**1.000**$^{\circ}$	0.542	0.467*	0.712	1.222*
DDIM inv.	**0.405***	0.559	0.874*	0.560	2.278	1.000°	0.552	0.412*	0.702	3.039*
SteerMusic	0.349*	0.632*	0.626*	**0.493**	1.980	0.920	**0.938**	0.616°	**0.571***	1.494*
Audio-Omni	0.234	**0.664***	0.610*	0.542	2.228	0.980	0.923	**0.631***	0.725	-0.852*

Instrument transfer	CLAP-T ↑	CLAP-A in-context ↑	KL ↓	FAD ↓	KAD ↓	PaSST^{E} ↑	Frame-F1^{P} ↑	chroma^{P} ↑	Audio Feat.^{E} ↓	isolation^{E} ↑
ProCraft (ours)	0.244	0.521	**0.504**	0.744	3.037	0.547	0.495	0.581	0.773	-1.960
ProCraft-plain	0.225*	0.505	0.546	**0.726**	**2.946**	0.607	0.497	0.603	0.791	-0.831
IMG (retrained)	0.192*	0.346*	1.070*	0.872	3.651	0.187*	0.191	0.131*	1.020*	-11.866*
ZETA	0.247	0.218*	1.319*	1.164	4.630	0.613	0.109	0.802*	1.024*	-0.014
DDIM inv.	0.264	0.209*	1.241*	1.116	4.377	**0.673***	0.095	**0.830***	1.103*	**6.943***
SteerMusic	**0.341***	**0.632***	0.852*	0.958	3.829	0.342*	**0.668**	0.391*	**0.720**	-11.137*
Audio-Omni	0.266°	0.580*	0.896*	0.940	3.756	0.613	0.582	0.722*	0.779	-0.457

Remove	CLAP-T ↑	CLAP-A in-context ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	edit rate^{E} ↑	edit strength^{E} ↑	Audio Feat.^{E} ↓
ProCraft (ours)	0.264	0.714	**0.546**	0.466	1.817	**-7.386**	1.154	2.493	0.660	12.911	0.568
ProCraft-plain	0.257	**0.718**	0.550	**0.440**	**1.725**	-7.786	**1.144**	**2.288**	0.660	11.956	0.552
IMG (retrained)	0.224*	0.613*	0.669*	0.644	2.737	-12.504*	1.273*	2.487	0.553°	7.443*	0.554
ZETA	0.172*	0.145*	2.964*	0.915	2.744	-46.183*	3.559*	7.650*	0.673	16.348°	0.791*
DDIM inv.	0.110*	0.087*	2.302*	0.916	2.951	-44.831*	3.256*	7.011*	**0.687**	**16.599**$^{\circ}$	0.827*
SteerMusic	**0.322***	0.628*	0.832*	0.597	2.386	-25.596*	1.619*	5.336*	0.287*	2.213*	**0.511**
Audio-Omni	0.244°	0.478*	0.665*	0.573	2.305	-24.875*	1.449*	3.730*	0.300*	1.281*	0.546

Extract	CLAP-T ↑	CLAP-A in-context ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	edit rate^{E} ↑	edit strength^{E} ↑	Audio Feat.^{E} ↓
ProCraft (ours)	0.280	0.440	**0.528**	0.671	2.534	**-15.725**	**1.851**	3.654	0.680	8.526	0.799
ProCraft-plain	0.273	0.468°	0.595°	0.667	2.577	-17.546*	1.935	**3.598**	**0.700**	**9.210**	0.816
IMG (retrained)	0.163*	0.352*	1.455*	0.835	3.424	-28.327*	2.410*	4.225*	0.067*	-14.241*	0.878°
ZETA	0.298	0.255*	1.828*	0.815	2.623	-38.028*	2.990*	6.732*	0.447*	-0.120*	0.824
DDIM inv.	0.228*	0.177*	1.878*	0.807	2.577	-41.259*	3.082*	6.515*	0.380*	-4.820*	0.873
SteerMusic	**0.314***	0.532*	0.900*	0.644	**2.456**	-27.196*	2.004°	5.480*	0.527*	2.907*	**0.665***
Audio-Omni	0.244*	**0.663***	0.632°	**0.628**	2.458	-23.980*	1.876	3.769	0.540°	4.287*	0.755

Denoise	CLAP-T ↑	CLAP-A in-context ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	Audio Feat.^{E} ↓
ProCraft (ours)	0.092	**0.509**	**0.661**	**0.503**	**1.948**	**-4.735**	**1.133**	**2.387**	0.660
ProCraft-plain	0.093	0.502	0.669	0.507	1.975	-4.972	1.179	2.394	0.648
IMG (retrained)	0.088	0.503	0.784*	0.594	2.488	-8.515*	1.269*	2.528	0.593
ZETA	0.180*	0.108*	1.118*	1.290	5.676	-35.418*	2.477*	6.306*	**0.520***
DDIM inv.	0.180*	0.166*	1.421*	1.211	5.407	-32.180*	2.562*	6.488*	0.638
SteerMusic	**0.230***	0.418*	0.986*	0.635	2.275	-29.896*	1.671*	5.293*	0.652
Audio-Omni	0.156*	0.438*	0.971*	0.625	2.482	-26.023*	1.594*	3.718*	0.660

Table 6: Combined edit and audio effects benchmark, **real audio** domain, 150 samples per task; the target is the edited and processed audio. E=editing, P=preservation, E+P=both. **Bold** = best in column, shaded worst to best.

Accompany	CLAP-T ↑	CLAP-A in-context ↑	KL ↓	FAD ↓	KAD ↓	PaSST^{E} ↑	APA^{P} ↑	StructDist^{P} ↓	Audio Feat.^{E} ↓	isolation^{E} ↑
ProCraft (ours)	0.303	0.483	0.959	0.256	0.852	0.693	0.366	123.701	0.846	3.873
ProCraft-plain	**0.316**	0.500°	**0.841**	0.260	0.884	**0.773**	0.461	123.593	0.839	**8.704***
IMG (retrained)	0.106*	0.491	1.879*	0.404	1.450	0.180*	0.000	113.529*	0.991*	-11.355*
ZETA	0.113*	0.594*	4.138*	0.387	1.104	0.287*	0.417	96.880*	1.089*	-13.347*
DDIM inv.	0.055*	0.435*	3.426*	0.464	1.590	0.200*	0.237	146.692*	1.002*	-17.029*
SteerMusic	0.234*	**0.630***	1.781*	0.348	1.159	0.153*	0.270	**83.552***	**0.764**	-13.333*
Audio-Omni	0.131*	0.517*	1.101°	**0.253**	**0.834**	0.613	**0.797**	123.593	0.876	-5.432*

Cover	CLAP-T ↑	CLAP-A in-context ↑	FAD ↓	KAD ↓	PaSST^{E} ↑	APA^{P} ↑	chroma^{P} ↑	isolation^{E} ↑
ProCraft (ours)	**0.308**	0.492	0.344	1.212	0.953	0.700	0.574	13.185
ProCraft-plain	0.295°	0.488	0.333	1.162	**0.980**	0.672	0.560	**14.948**
IMG (retrained)	0.196*	0.488	0.326	1.168	0.267*	0.301	0.184*	-10.217*
ZETA	0.133*	**0.613***	0.319	**0.949**	0.540*	0.835	0.473*	-4.222*
DDIM inv.	-0.004*	0.461°	0.472	1.638	0.460*	0.712	0.328*	-9.282*
SteerMusic	0.278°	0.579*	**0.302**	0.964	0.687*	0.806	0.517*	-5.800*
Audio-Omni	0.268*	0.585*	0.344	1.245	0.947	**0.899**	**0.682***	-4.701*

Instrument transfer	CLAP-T ↑	CLAP-A in-context ↑	FAD ↓	KAD ↓	PaSST^{E} ↑	chroma^{P} ↑	isolation^{E} ↑
ProCraft (ours)	0.216	0.368	0.641	2.466	0.547	0.451	-0.651
ProCraft-plain	0.218	0.387	**0.602**	**2.259**	0.587	0.527*	**2.435**
IMG (retrained)	0.166*	0.381	0.825	3.409	0.147*	0.134*	-12.336*
ZETA	0.188	0.177*	0.938	2.903	0.367*	0.508	-7.554*
DDIM inv.	0.145*	0.149*	0.953	3.223	0.307*	0.382	-16.860*
SteerMusic	**0.300***	**0.693***	0.683	2.316	0.047*	0.426	-23.328*
Audio-Omni	0.264*	0.580*	0.661	2.565	**0.613**	**0.646***	-6.753*

Remove	CLAP-T ↑	CLAP-A in-context ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	edit rate^{E} ↑	edit strength^{E} ↑	Audio Feat.^{E} ↓
ProCraft (ours)	0.228	0.685	0.815	0.351	1.176	-12.112	1.183	2.418	0.507	8.793	0.616
ProCraft-plain	0.222	**0.713**°	**0.795**	**0.334**	**1.125**	**-11.491**	**1.136**	**2.361**	0.407°	7.221	0.616
IMG (retrained)	0.208°	0.455*	1.110*	0.586	2.397	-20.292*	1.359*	2.686*	0.447	7.415	0.557°
ZETA	0.141*	0.117*	3.463*	0.956	2.962	-47.959*	2.981*	6.939*	**0.787***	**21.095***	0.766*
DDIM inv.	0.117*	0.081*	2.960*	0.918	2.997	-45.645*	2.853*	6.675*	0.787*	20.177*	0.768*
SteerMusic	**0.277***	0.596*	0.985*	0.425	1.397	-27.321*	1.695*	5.398*	0.193*	2.311*	**0.508***
Audio-Omni	0.242	0.467*	1.005*	0.500	1.980	-26.489*	1.627*	3.046*	0.369°	7.119	0.610

Extract	CLAP-T ↑	CLAP-A in-context ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	edit rate^{E} ↑	edit strength^{E} ↑	Audio Feat.^{E} ↓
ProCraft (ours)	0.273	0.454	0.742	0.551	1.869	**-19.038**	1.764	3.328	0.767	8.590	0.810
ProCraft-plain	**0.291***	0.452	**0.702**	0.539	1.880	-21.592*	1.783	3.305	**0.787**	**10.884**°	0.776
IMG (retrained)	0.157*	0.303*	1.699*	0.779	3.167	-32.572*	2.507*	4.379*	0.040*	-12.499*	0.949*
ZETA	0.243	0.162*	2.445*	0.815	2.392	-39.465*	2.779*	6.568*	0.453*	0.130*	0.785
DDIM inv.	0.196*	0.117*	2.386*	0.837	2.538	-42.502*	2.996*	6.556*	0.473*	-0.411*	0.774
SteerMusic	0.278	0.501*	1.083*	**0.447**	**1.440**	-29.762*	1.864	5.413*	0.527*	2.282*	**0.587***
Audio-Omni	0.245*	**0.595***	0.718	0.522	1.765	-25.364*	**1.681**	**3.227**	0.733	6.426	0.747

Denoise	CLAP-T ↑	CLAP-A in-context ↑	KL ↓	FAD ↓	KAD ↓	SI-SDR^{E+P} ↑	mel-L1^{E+P} ↓	MRSTFT^{E+P} ↓	Audio Feat.^{E} ↓
ProCraft (ours)	0.081	0.502	**0.815**	0.435	1.458	**-10.141**	1.255	2.587	0.793
ProCraft-plain	0.081	**0.521**	0.854	**0.403**	**1.352**	-10.370	**1.199**	**2.524**	0.778
IMG (retrained)	0.056°	0.409*	1.130*	0.557	2.243	-16.107*	1.462*	2.939*	0.695°
ZETA	0.175*	0.047*	3.430*	0.968	2.995	-50.297*	4.169*	8.783*	1.236*
DDIM inv.	0.147*	-0.001*	2.640*	0.963	3.345	-49.838*	3.291*	6.866*	0.893
SteerMusic	**0.222***	0.409*	1.241*	0.520	1.698	-32.832*	1.817*	5.531*	**0.663***
Audio-Omni	0.160*	0.408*	1.188*	0.651	2.596	-28.934*	1.913*	3.520*	0.819

D TRAINING AND INFERENCE SETTINGS

We train for 20,000 steps with a 1,000-step linear warmup into cosine decay. The optimizer is AdamW with betas 0.9 and 0.95, weight decay 0.01, and gradient clipping at 1.0. Peak learning rates are 5×10^{-5} for the backbone and 3×10^{-4} for projectors and adapters, with an effective batch of 16 sequences in bf16 and uniform instruction dropout of 0.15 as the classifier-free-guidance null signal. At inference the model encodes the instruction, source codes, and query embeddings once, then samples the target at temperature 1.0 with CFG scale = 1.5.

E FULL METRIC DEFINITIONS

We provide the exact definitions and implementation details for all metrics reported in the main tables. The notation follows Sections 2 and 4.1: x denotes the source clip, c the instruction, $\hat{y}$ the system output, and y the target when one is available.

AFx-Rep cosine (Editing, audio effects). We compute Audio Effects Representation (AFx-Rep) cosine similarity using the representation released with ST-ITO (Steinmetz et al., 2024). Let $f_{\text{AFx}}(\cdot)$ denote its encoder. We compute

$$\text{AFxRep}(\hat{y},y) = \frac{f_{\text{AFx}}(\hat{y})^\top f_{\text{AFx}}(y)}{\|f_{\text{AFx}}(\hat{y})\|_2 \|f_{\text{AFx}}(y)\|_2},$$

where y is the ground-truth signal processed with the requested effect chain. Because AFx-Rep is trained to characterize the applied effect chain rather than musical content, the cosine measures whether the requested treatment is present in the output.

Audio-feature distance (Editing). We compare seven audio features of the output and target, adapted from the audio-feature loss of Diff-MST (Vanka et al., 2024). That loss compares RMS level, crest factor, stereo width, stereo imbalance, and a Bark spectrum. Because our audio is mono, we keep RMS level and crest factor and drop the two stereo features. We replace the Bark spectrum with five scalar spectral features: centroid, bandwidth, 85% rolloff, flatness, and contrast. We compute all seven features on the mono signal from a 2048-point STFT with a hop of 512 samples. RMS level, crest factor, and spectral contrast are measured in dB. Centroid, bandwidth, and rolloff are measured in octaves above 20 Hz. Let $f_k(z)$ denote feature k of signal z and a_k its scaling constant. The distance is

$$D_{\text{feat}}(\hat{y},y) = \frac{1}{7}\sum_{k=1}^{7} \frac{|f_k(\hat{y}) - f_k(y)|}{a_k}.$$

The equal weights and the scaling constants differ from the loss weights of Diff-MST. We set $a_k = 6\,\text{dB}$ for the three features in dB and $a_k = 1$ octave for centroid, bandwidth, and rolloff. For spectral flatness, $a_k = 0.15$. These values give the seven terms comparable magnitude. Both signals are RMS-normalized before feature extraction, so the RMS-level term is close to zero. The metric measures whether the requested audio effects characteristics were reached, providing better observation than waveform-level reference metrics, which may be overly sensitive to content differences.

CLAP-A vs. source. We compute Contrastive Language–Audio Pretraining audio–audio similarity (CLAP-A) between the output and source:

$$\text{CLAP-A}(\hat{y},x) = \frac{f_{\text{audio}}(\hat{y})^\top f_{\text{audio}}(x)}{\|f_{\text{audio}}(\hat{y})\|_2 \|f_{\text{audio}}(x)\|_2}.$$

CLAP-T out (pred vs. instr). We use Contrastive Language–Audio Pretraining text–audio similarity (CLAP-T) (Wu et al., 2023) to measure output-to-instruction adherence. Specifically,

$$\text{CLAP-T}(\hat{y},c) = \frac{f_{\text{audio}}(\hat{y})^\top f_{\text{text}}(c)}{\|f_{\text{audio}}(\hat{y})\|_2 \|f_{\text{text}}(c)\|_2}.$$

Edit rate and edit strength (Editing; remove and extract). We first separate each signal into instrument stems using HTDemucs-6s (Rouard et al., 2023). Let

$$\mathscr{E}_t(z) = \sqrt{\frac{1}{T}\sum_{n=1}^{T} s_t(z)[n]^2}$$

denote the RMS energy of target-instrument stem t in signal z, where $s_t(z)$ is the corresponding HTDemucs-6s stem. Remove and extract strength are respectively

$$S_{\text{remove}} = 10\log_{10}\frac{\mathscr{E}_t(x)}{\mathscr{E}_t(\hat{y})}, \qquad S_{\text{extract}} = 10\log_{10}\frac{\mathscr{E}_t(\hat{y})}{\sum_{s\neq t}\mathscr{E}_s(\hat{y})},$$

with both clipped to $[-30, 30]$ dB. Edit rate thresholds the corresponding quantities at the example level:

$$\text{ER}_{\text{remove}} = \mathbb{1}\left[\mathscr{E}_t(\hat{y}) \leq \tfrac{1}{2}\mathscr{E}_t(x)\right], \qquad \text{ER}_{\text{extract}} = \mathbb{1}\left[\mathscr{E}_t(\hat{y}) > \sum_{s\neq t}\mathscr{E}_s(\hat{y})\right].$$

The reported edit rate is the mean of these binary indicators over examples. These metrics ask whether the requested stem actually changed rather than how closely the output matches a particular reference. Because HTDemucs-6s supplies the stems used by the metric, it acts as the judge and is not scored on these rows.

FAD/KAD (output naturalness). We compute Fréchet Audio Distance (FAD) (Kilgour et al., 2019) and Kernel Audio Distance (KAD) (Chung et al., 2025) on CLAP-LAION-music embeddings against the same fixed real-music reference pool for every system, following Gui et al. (2024). Let (μ_P, Σ_P) and (μ_Q, Σ_Q) denote the Gaussian statistics of the output and reference embeddings, respectively. FAD is

$$\text{FAD} = \|\mu_P - \mu_Q\|_2^2 + \text{Tr}\left(\Sigma_P + \Sigma_Q - 2(\Sigma_P\Sigma_Q)^{1/2}\right).$$

KAD uses the kernel-based maximum mean discrepancy estimator of Chung et al. (2025). Lower FAD and KAD indicate distributions closer to the real-music reference pool. We use the same fixed reference pool for every system.

Frame-F1 (Preservation, instrument transfer). We compute frame-level pitch F1 (Frame-F1) between the output's MuScriptor transcription and the source-track Musical Instrument Digital Interface (MIDI) ground truth at 75 frames per second. We accumulate true positives (TP), false positives (FP), and false negatives (FN) over frames and compute

$$P = \frac{\text{TP}}{\text{TP}+\text{FP}}, \qquad R = \frac{\text{TP}}{\text{TP}+\text{FN}}, \qquad \text{Frame-F1} = \frac{2PR}{P+R}.$$

We use Frame-F1 only for instrument transfer, where content is preserved by construction, and only on Slakh, since the real-domain evaluation set has no MIDI ground-truth.

Isolation strength (Editing; combined benchmark, accompany, cover and instrument transfer). Using the RMS stem energy $\mathscr{E}_t(\cdot)$ defined above, isolation strength applies the extract form of edit strength to the output:

$$S_{\text{iso}} = 10\log_{10}\frac{\mathscr{E}_t(\hat{y})}{\sum_{s\neq t}\mathscr{E}_s(\hat{y})}.$$

Higher values indicate that the requested instrument accounts for more of the separated output energy relative to other instruments. The metric is computed from separated stems using HTDemucs-6s as the separator; consequently, HTDemucs-6s itself is not scored on this metric.

KL (output naturalness). We compute the Kullback–Leibler (KL) divergence between the PaSST (Koutini et al., 2022) audio-tagging posterior distributions of the output and target. If p_y and $p_{\hat{y}}$ denote their posterior vectors, respectively,

$$D_{\text{KL}}(p_y \| p_{\hat{y}}) = \sum_k p_y(k)\log\frac{p_y(k)}{p_{\hat{y}}(k)}.$$

Lower KL indicates that the output induces an audio-tag distribution closer to that of the target.

Melody chroma accuracy (Preservation; cover and instrument transfer). Following the cover-generation evaluation of Tan et al. (2024b), we use melody chroma accuracy defined in `mir_eval` (Raffel et al., 2014) to compare the estimated melody of the output with that of the source frame by frame. For frames in which both estimates are voiced, a prediction is counted as correct when the estimated pitches agree modulo octave within the chroma tolerance used by `mir_eval`. Melody

chroma accuracy is the fraction of reference voiced frames satisfying this criterion. It therefore measures whether the re-voiced output follows the source melody irrespective of octave. An unchanged copy of the input scores highly by construction, so we interpret this metric together with the Editing metrics rather than in isolation.

PaSST instrument match (Editing). PaSST (Koutini et al., 2022) predicts AudioSet logits over instrument-related classes. We map the highest-scoring class to our instrument families and report the classification accuracy against the instrument family requested by the instruction. We skip systems for which the judge's own separation determines the classification result, as well as systems that always emit a single fixed instrument family.

SI-SDR, mel-L1 and MRSTFT vs. ground-truth (Editing and Preservation). We compute all three metrics against deterministic ground-truth targets. Scale-Invariant Signal-to-Distortion Ratio (SI-SDR) is used for remove, extract, and denoise, while mel-spectrogram L1 distance (mel-L1) and Multi-Resolution Short-Time Fourier Transform loss (MRSTFT) additionally evaluate audio effects against the processed target. For zero-mean reference y and estimate $\hat{y}$, SI-SDR is

$$\text{SI-SDR}(y,\hat{y}) = 10\log_{10}\frac{\|\alpha y\|_2^2}{\|\hat{y}-\alpha y\|_2^2}, \qquad \alpha = \frac{\langle \hat{y}, y\rangle}{\|y\|_2^2}.$$

Let $\mathbf{M}(z)$ denote the log-mel spectrogram of signal z. The mel-L1 distance is

$$\mathscr{L}_{\text{mel}} = \frac{1}{N}\left\|\mathbf{M}(\hat{y}) - \mathbf{M}(y)\right\|_1,$$

where N is the number of time–frequency bins. For each Short-Time Fourier Transform (STFT) resolution r, with magnitude spectrogram $\mathbf{S}_r(z)$, we compute spectral convergence and log-magnitude error as

$$\mathscr{L}_{\text{SC}}^{(r)} = \frac{\|\mathbf{S}_r(y) - \mathbf{S}_r(\hat{y})\|_F}{\|\mathbf{S}_r(y)\|_F}, \qquad \mathscr{L}_{\log}^{(r)} = \frac{1}{N_r}\left\|\log(\mathbf{S}_r(y)+\varepsilon) - \log(\mathbf{S}_r(\hat{y})+\varepsilon)\right\|_1.$$

MRSTFT is

$$\mathscr{L}_{\text{MRSTFT}} = \sum_{r\in\{2048,1024,512,256\}} \left(\mathscr{L}_{\text{SC}}^{(r)} + \mathscr{L}_{\log}^{(r)}\right).$$

All three metrics are computed on mono audio at 48 kHz after RMS level matching the system output to the target.

StructDist and APA (Preservation; accompany and cover). Structural distance (StructDist) compares the temporal self-similarity structure of the output and source. Let $\mathbf{H}(z) \in \mathbb{R}^{T\times D}$ denote frame-level MERT representations of signal z, with self-similarity matrix

$$\mathbf{A}(z)_{ij} = \cos(\mathbf{H}(z)_i, \mathbf{H}(z)_j).$$

StructDist is the RMS distance between the two self-similarity matrices:

$$\text{StructDist}(\hat{y},x) = \sqrt{\frac{1}{T^2}\left\|\mathbf{A}(\hat{y}) - \mathbf{A}(x)\right\|_F^2}.$$

It is computed directly on the generated output to measure the structure distance between the output and the source.

Accompaniment Prompt Adherence (APA) (Grachten & Nistal, 2025) measures how well generated accompaniments fit their source-audio prompts at the distribution level. Let C be the candidate set of context–stem pairs, R a reference set of matched context–stem pairs, and R' a reference set with randomly mismatched stems. APA is defined from their Fréchet Audio Distances (FADs) as

$$\text{APA} = \frac{1}{2} + \frac{\text{FAD}_{C,R'} - \text{FAD}_{C,R}}{2\,\text{FAD}_{R,R'}},$$

and is clipped to $[0,1]$. A score of 0.5 indicates that the candidate distribution is equally distant from the matched and mismatched references, while higher values indicate stronger prompt adherence. We apply APA to accompany and cover.

F DATA PIPELINE DETAILS

Corpus and instrument families. The training corpus is Slakh2100 multitrack stems at the standard splits, with stems summed into General MIDI families. Training uses eight families. Four are the rhythm section and keys: bass and drums, guitar and piano, and the other four are sustained voices: organ and brass, reed and ensemble. Every edit targets one family. Input and target windows are 30 seconds at 48 kHz on disk, downmixed to 24 kHz mono for the codec. A window is randomly selected from each audio sample and will be resampled if any target stem is more than 70% near-silent or peaks below 0.10. We skip an audio sample if there's no valid window in 10 attempts.

Effect authoring and validation. We build two things through one LLM call: prose a producer might say, and the audio effects settings that realize it. The prompt offers 3 to 5 audio effect families sampled from the library, and one of the target mixing styles to aim for. Each of them is drawn from a predefined pool. Sampling the style per item keeps the corpus from collapsing onto a few favored effects, and rotating the examples keeps the model from copying them back. Chains then execute on the stems, with stereo and spatial effects banned because the evaluation is mono, and each chain's output is RMS-matched to its input so the volume doesn't change while effects are applied.

Cover targets. The cover reduction collects pitched notes from all non-drum stems, applies salience-based deletion with octave reduction, and renders onto one wide-range target instrument, including piano, guitar, organ, and ensemble.

Instrument transfer and event tasks. Instrument-transfer items render one pitched track's MIDI from the source and a target instrument sound font, so the pitch is unchanged, and only instrument identity differs. Denoise items mix one environmental sound into the music, drawn from FSD50K in training and from ESC-50 in the test sets. An ambient sound such as rain or wind is looped to fill the clip, at 9 to 18 dB below the music. A transient sound such as a dog bark or car horn is repeated 3 to 6 times at random positions across the clip, at 3 to 12 dB below the music.

G MIXING MOTIVATIONS AND EXAMPLE INSTRUCTIONS

G.1 MIXING MOTIVATIONS

Each item that carries audio effects samples one of the 30 mixing motivations below uniformly at random. The language model receives it as the aesthetic that the audio effects and the abstract description should aim for; the arrangement part of the instruction stays literal. Items without audio effects do not use a motivation.

90s grunge	folk back-porch	reggae dub
ambient and dreamy	gospel warmth	shoegaze wall-of-sound
arena rock	gritty electronic	smoky jazz club
boom-bap hip-hop	industrial cold	spaghetti-western twang
bossa nova café	lo-fi bedroom	surf rock
chillwave sunset	modern radio pop	synthwave neon
cinematic and widescreen	motown groove	trap night-drive
crisp and hi-fi	orchestral film score	vintage soul
drum-and-bass rollers	psychedelic haze	warm analog tape
dubby and cavernous	raw garage punk	witch-house murk

Table 7: The 30 mixing motivations, in alphabetical order.

G.2 EXAMPLE INSTRUCTIONS

Table 8 shows instructions from our demo page, covering all seven operations and requests that combine an operation with audio effects. Source audio, model outputs, and baseline comparisons are available at https://anonymous5566mir.github.io/ICLR2027/.

Task	Instruction
Single operations	
Remove	Cut the bass from the mix
Extract	Extract only the guitar track.
Accompany	Generate a piano part that complements the track
Cover	Arrange entire track as solo piano performance
Instrument transfer	Replace guitar with bass using same notes.
Denoise	Eliminate all non-music sounds.
Audio effects	Add rich tube saturation and hall reverb to the piano.
Combined editing and audio effects	
Remove	Remove the bass. Add pumping compression to the piano, airy high-pass filter to the drums, and warm saturation to the guitar.
Extract	Extract the bass. Add punchy compression with a subtle saturation to the bass.
Accompany	Create a standalone piano part that complements the track. Add gritty bitcrush with analog noise.
Cover	Reconfigure full track for solo piano performance. Put on tamed low-end, boosted mid-range, tight delay.
Instrument transfer	Swap bass for piano, keep notes identical. Add gritty lo-fi bit crush with noise floor.
Denoise	Eliminate the non-musical rain audio. Apply subtle bitcrush haze, soft saturation, gentle noise floor.

Table 8: Example free-form instructions from the demo page.

H FUNCTIONALITY COVERAGE JUDGMENTS

Table 1 distinguishes native support from looser uses of each system; the following judgments explain its cells. A model is able (●) on a functionality when that functionality is a native and reliable mode. It is partial (▲) when the capability is indirect, out of distribution, quality-limited, or a loose application. It is unable (×) when the model has no such mode.

Separators. AudioSep isolates a described source, so extract is native and remove is partial, needing a query-then-subtract step that empirically guts dense music mixes. Denoise is partial because AudioSep targets environmental events rather than the complete editing task, and free phrasing is partial because it accepts target names rather than edit instructions. SD-Codec routes by domain rather than instrument: it can denoise by decoding the music stream, but extracting that whole stream is only a partial match to instrument extraction, and it cannot remove a named instrument.

Stem imputers. MSDM and MusicGen-Stem generate a stem consistent with the others, so accompany is native. MSDM has no text conditioning, so its free phrasing is unable. MusicGen-Stem is a text-to-music model whose edits are driven by stem conditioning, so its free phrasing is partial.

Cover and instrument transfer systems. Pop2Piano and PiCoGen2 generate solo-piano covers, so Table 1 marks cover as native and other operations as unsupported; neither takes a free-text instruction. AP-Adapter supports instrument transfer, cover, and accompaniment, but its timbre edit changes the whole clip rather than a named part, so the corresponding stem-level tasks are partial matches even though it accepts free-form target text. MuseControlLite conditions on melody and reference audio to transfer a whole clip's instrument; we therefore mark instrument transfer as supported and accompany and cover as partial.

Instruction and inversion editors. Instruct-MusicGen covers remove, extract, and accompany from a fixed Title-case template, so those three are able. It's not trained on cover, instrument transfer, or denoise. Its templated prompt is not freely phrased text and it has no audio effect vocabulary, so those cells are unable. Audio-Omni (Tian et al., 2026) is a freely phrased instruction editor; in our listening test it is competitive on accompany, cover, and instrument transfer. It falls far behind on remove and denoise, and audio effects are unsupported. ZETA edits from a target caption, so free phrasing is able. Instrument transfer is able through the target caption, cover is partial, remove is a partial coarse nudge, and audio effects are a partial non-parametric text request. Surgical extract, accompany, and denoise are unable.

LLM2Fx-Tools maps language to executable effect chains but does not perform any other edits. ProCraft covers every column because it reads free-form instructions and performs each evaluated edit including audio effects.

I EDIT-HEAD PROBING PROTOCOL

Features. For each item, we run ProCraft once on the input sequence, made of the instruction tokens, the WavTokenizer codes of the 30,s source clip, and the 32 learnable query embeddings. We take the edit head h_{edit}, the last hidden state at the 32 query positions, and average it over those positions into one 896-dimensional vector.

Classifier. We use an MLP with one hidden layer of 128 units, trained on standardized inputs for at most 2000 iterations. Accuracy is computed over the pooled out-of-fold predictions of 5-fold stratified cross-validation, with shuffled folds and a fixed seed. Chance is the frequency of the most common class. Classes with fewer than three items would be dropped, but none of the four probes below has one.

Operation and target instrument. We build 350 edits on the Slakh test split with the pipeline of Section 3, using all eight instrument families. The set has 50 edits for each of the seven tasks: remove, extract, accompany, cover, instrument transfer, denoise, and audio effects. The operation probe predicts the task of all 350 edits (7 classes, chance 0.143). The target-instrument probe uses the 299 edits whose instruction names an instrument, which excludes the 50 denoise edits and one audio effect edit. Its label is the family that is removed, extracted or added, the cover instrument, the destination of an instrument transfer, or the processed track of an audio effect edit. The eight classes hold between 18 and 49 items, and chance is 0.164.

Instrument of a single stem. We cut 400 clips from isolated Slakh stems, 50 per family (chance 0.125). Each clip is the loudest 30,s window of its stem, searched with a 5,s hop and required to be active in at least 80% of its seconds. We take at most two stems per track and family, drawing 352 clips from the test split and 48 from the validation split where a family ran short.

Pitch of a single note. We render 343 clips with FluidSynth and the FluidR3 General MIDI soundfont: one for each of the seven pitched families and each MIDI pitch from 36 to 84 (C2 to C6). This gives 49 classes with 7 clips each (chance 0.020). Each clip is 30,s long and repeats one note at velocity 96, sounding for 1.5,s every 2,s, on the family's General MIDI program.

Baselines. The baselines train the same classifier on the mean of the checkpoint's input embeddings for the same items. The two instruction properties use the embeddings of the non-padding instruction tokens, and the two audio properties use the embeddings of the source codec tokens.

J LISTENING TEST PROTOCOL

The test follows the MUSHRA layout (ITU-R, 2015) without anchors, since no gold output exists for the generative tasks. We selected three source-recording-and-instruction pairs per task from the real combined benchmark. For those eighteen pairs, we collected the outputs of ProCraft, ZETA, DDIM inversion, SteerMusic and Audio-Omni. We trim every clip to 30 s and loudness-normalise it to -23 LUFS with a -1 dBTP limit, then fade it over 10 ms and encode it at 192 kbps. Each rater receives one randomly chosen recording-and-instruction pair per task and completes the six trials in random order. In each trial the rater sees the five stimuli labelled A to E in a random permutation and rates them on three attributes in turn. Editing asks whether the change the instruction names was made, arrangement and audio effects; Preservation asks whether everything the instruction did not name is intact; audio quality asks for freedom from artifacts, independent of correctness. Task-specific guidance under each question states what Editing and Preservation mean for that task. The reference is the source recording with the instruction, and raters must start every clip before advancing; the player switches between stimuli at the same playhead position. Raters answered a questionnaire on musical training and production experience; the sixteen raters reported between no and more than ten years of training, fourteen of them three years or more, and seven produce music as a hobby. Table 9 lists every cell mean behind Figure 6.

Table 9: Listening test, mean rating (0 to 5) per task, attribute and system over 16 raters. **Bold** = highest sample mean; underline = second-highest sample mean; significance is reported in Table 11.

Task	Attribute	ProCraft (ours)	Audio-Omni	SteerMusic	ZETA	DDIM inv.
Remove	Editing	**2.50**	1.69	2.00	0.81	0.81
	Preservation	**4.06**	1.38	2.88	0.44	0.50
	Audio quality	**3.56**	0.69	2.12	1.25	1.38
Extract	Editing	3.12	**3.25**	2.38	0.56	0.06
	Preservation	3.44	**3.69**	3.12	0.31	0.12
	Audio quality	3.06	**3.69**	2.75	1.31	0.31
Accompany	Editing	2.19	**2.31**	1.88	2.25	2.12
	Preservation	1.88	**3.81**	2.69	0.44	0.44
	Audio quality	2.38	**3.19**	1.81	2.25	2.44
Cover	Editing	2.06	**3.19**	2.81	2.06	1.44
	Preservation	1.31	3.06	**3.38**	1.19	0.75
	Audio quality	1.81	**2.62**	2.50	2.44	1.50
Instrument transfer	Editing	**3.12**	3.06	1.12	1.56	0.31
	Preservation	3.38	**3.50**	2.00	1.44	0.31
	Audio quality	3.25	**3.62**	2.25	2.44	0.94
Denoise	Editing	**3.56**	2.00	0.69	0.00	0.69
	Preservation	**3.88**	2.19	2.62	0.06	0.19
	Audio quality	**3.75**	2.06	1.88	0.62	0.56

Statistical tests. Because each rater evaluated all five systems on all six tasks and three attributes, we fit a repeated-measures ANOVA separately for each task, with system and attribute as within-rater factors (Table 10). We report partial eta squared for each effect. Every task has a significant system-by-attribute interaction, so we follow each ANOVA with paired t-tests comparing ProCraft with each of the four alternatives on each attribute. Holm's step-down procedure corrects the twelve comparisons within each task; Table 11 lists all comparisons, including those that are not significant, and the analysis output retains their confidence intervals.

Table 10: Repeated-measures ANOVA on the listening ratings, one per task, with system (5 levels) and attribute (3 levels) as within-rater factors over 16 raters. η_p^2 = partial eta squared.

Task	Effect	F (df)	p	η_p^2
Remove	system	19.31 (4, 60)	<0.001	0.56
	attribute	4.39 (2, 30)	0.021	0.23
	system × attribute	6.25 (8, 120)	<0.001	0.29
Extract	system	62.83 (4, 60)	<0.001	0.81
	attribute	3.06 (2, 30)	0.062	0.17
	system × attribute	2.66 (8, 120)	0.010	0.15
Accompany	system	4.92 (4, 60)	0.002	0.25
	attribute	2.46 (2, 30)	0.103	0.14
	system × attribute	11.34 (8, 120)	<0.001	0.43
Cover	system	9.71 (4, 60)	<0.001	0.39
	attribute	1.95 (2, 30)	0.160	0.11
	system × attribute	4.15 (8, 120)	<0.001	0.22
Instrument transfer	system	29.75 (4, 60)	<0.001	0.66
	attribute	10.59 (2, 30)	<0.001	0.41
	system × attribute	2.17 (8, 120)	0.034	0.13
Denoise	system	52.35 (4, 60)	<0.001	0.78
	attribute	5.48 (2, 30)	0.009	0.27
	system × attribute	3.98 (8, 120)	<0.001	0.21

Table 11: All ProCraft–baseline follow-up comparisons after the task-wise ANOVAs. Each cell gives the mean rating difference (ProCraft minus baseline) and Holm-adjusted p; correction covers 12 comparisons per task.

Task	Attribute	Audio-Omni	SteerMusic	ZETA	DDIM inv.
Remove	Editing	+0.81 (0.341)	+0.50 (0.376)	+1.69 (0.047)	+1.69 (0.047)
	Preservation	+2.69 (<0.001)	+1.19 (0.014)	+3.62 (<0.001)	+3.56 (<0.001)
	Audio quality	+2.88 (<0.001)	+1.44 (0.005)	+2.31 (0.004)	+2.19 (0.007)
Extract	Editing	-0.12 (0.791)	+0.75 (0.208)	+2.56 (<0.001)	+3.06 (<0.001)
	Preservation	-0.25 (0.656)	+0.31 (0.656)	+3.12 (<0.001)	+3.31 (<0.001)
	Audio quality	-0.62 (0.328)	+0.31 (0.656)	+1.75 (0.019)	+2.75 (<0.001)
Accompany	Editing	-0.12 (1.000)	+0.31 (1.000)	-0.06 (1.000)	+0.06 (1.000)
	Preservation	-1.94 (0.071)	-0.81 (1.000)	+1.44 (0.052)	+1.44 (0.052)
	Audio quality	-0.81 (1.000)	+0.56 (1.000)	+0.12 (1.000)	-0.06 (1.000)
Cover	Editing	-1.12 (0.090)	-0.75 (0.660)	+0.00 (1.000)	+0.62 (1.000)
	Preservation	-1.75 (0.002)	-2.06 (<0.001)	+0.12 (1.000)	+0.56 (0.754)
	Audio quality	-0.81 (0.338)	-0.69 (0.810)	-0.62 (0.819)	+0.31 (1.000)
Instrument transfer	Editing	+0.06 (1.000)	+2.00 (<0.001)	+1.56 (0.001)	+2.81 (<0.001)
	Preservation	-0.12 (1.000)	+1.38 (0.014)	+1.94 (0.002)	+3.06 (<0.001)
	Audio quality	-0.38 (0.755)	+1.00 (0.090)	+0.81 (0.090)	+2.31 (0.004)
Denoise	Editing	+1.56 (0.003)	+2.88 (<0.001)	+3.56 (<0.001)	+2.88 (<0.001)
	Preservation	+1.69 (0.003)	+1.25 (0.010)	+3.81 (<0.001)	+3.69 (<0.001)
	Audio quality	+1.69 (0.008)	+1.88 (0.008)	+3.12 (<0.001)	+3.19 (<0.001)

K BASELINE PROTOCOLS

Instruct-MusicGen. The model fine-tunes a frozen MusicGen (Copet et al., 2023) with a LoRA (Hu et al., 2022) text-fusion module and a zero-initialized audio-fusion gate, and performs stem addition, removal, and extraction from templated instructions (Zhang et al., 2025b). We use the community-retrained checkpoint with MusicGen's default guidance of 3.0 and always provide its native training template. The chunked rows use its native 5-second windows with 1-second overlap-add reconstruction, for remove and extract. We do run it on accompany, where it returns a full mix rather than an isolated part, so we recover the added part with HTDemucs-6s before scoring and it is compared to the other systems on that separated stem. Instrument match is left unscored for it there, since the separator rather than the model is what isolates the part.

Instruct-MusicGen (retrained). We also retrain the same architecture on our corpus, following the released recipe except where the evaluation protocol requires a change. We keep the released adapter configuration, rank-32 LoRA with α 64 on the cross-attention query and value projections, the cosine schedule to 10^{-6}, and an effective batch of 32 sequences, and we train for the 20K steps ProCraft receives. The retraining differs from the released setup in three ways. Clips are 30 seconds instead of the released 5, since every test item in this paper is 30 seconds. The model sees all seven tasks in one run instead of the three its recipe targets, and the corpus covers eight instrument families instead of four. This comparison tests whether the architecture can handle the longer clips, wider instrument set and seven-task mixture used by ProCraft.

AudioSep. We use its official weights and resample the text-queried 32 kHz mask into our pipeline. For remove, we subtract the separated stem from the mix; for denoise, we query the noise and subtract it.

MSDM and MusicGen-Stem. Both task-specific systems output an accompaniment stem. MSDM runs its released Slakh source-imputation checkpoint over the four bass, drums, guitar and piano families at 22 kHz, generating the withheld part conditioned on the others. MusicGen-Stem runs the official `multistem` branch weights at 32 kHz, prompted to add the requested part over the source; its stem vocabulary is bass, drums and other, so guitar and piano both fall in the last. Both emit an isolated stem, as ProCraft does, so both can be scored on accompaniment metrics that a full-mix output would break.

Pop2Piano and PiCoGen2. We use their official weights and deterministic decoding. Both systems always emit solo piano, so an instrument-fidelity metric would score them perfectly on every item; those cells are masked.

AP-Adapter and MuseControlLite. We run AP-Adapter with `ap_scale` 0.5, 50 steps and guidance 7.5, and MuseControlLite with its defaults. We score both with per-conditioning alignment.

DDPM-inversion. This zero-shot editor applies the inversion method of Manor & Michaeli (2024) to a Stable Audio prior (Evans et al., 2025). It inverts the source under the caption "solo ⟨source⟩ playing" and replays the inversion noise under "solo ⟨target⟩" to change the instrument without reference audio.

SD-Codec (Bie et al., 2025)**.** We use its official weights and denoise by decoding the music quantizer once at 16 kHz, then resampling the result into our pipeline. Its CLAP and FAD cells reflect that bandwidth limit.

TokenSynth (Kim et al., 2025)**.** We use TokenSynth's official weights and give it the ground-truth pianoroll plus a target-instrument exemplar taken from a different item. It therefore reads the score and hears the timbre it is asked for, whereas ProCraft is given text alone, and it stands as a skyline rather than a like-for-like comparison.

FxSearcher (Ki et al., 2026) **and Text2FX** (Chu et al., 2025)**.** Both baselines use their official repositories to fit effect parameters on the source against a CLAP objective. Their DSP processing preserves the untouched audio by construction.

InstructFX2FX (Yu et al., 2026)**.** Its language model reads the instruction and picks the effect chain, and its optimizer fits the parameters against a CLAP objective. The planner is the same Qwen3 endpoint that wrote the corpus instructions.

ZETA, DDIM inversion, and SteerMusic. We run these zero-shot editors over the AudioLDM2-music prior (Liu et al., 2024) in their native caption-conditioned form. A source caption describes the mix, and a target caption states the arrangement and effects request. ZETA uses edit-friendly DDPM inversion, the DDIM row uses standard DDIM inversion with regeneration, and SteerMusic uses its delta-denoising edit.

HTDemucs-6s (Rouard et al., 2023)**.** We run the official six-source weights once per clip, dropping the requested stem for remove and keeping it for extract. It is also the judge that measures edit rate, edit strength and isolation strength, so those three cells are left blank for it: a system cannot be scored by itself.

Banquet (Watcharasupat & Lerch, 2024)**.** We run the official weights with the instrument name as query, in one pass per clip at the model's native rate. Its single-decoder separation copies samples from the input rather than generating new audio, as HTDemucs-6s does.

Audio-Omni (Tian et al., 2026)**.** We run this free-form instruction editor with its official weights and default decoding on the same combined instructions as ProCraft. It provides a direct instruction-following comparison alongside the caption-conditioned inversion editors in Table 5.